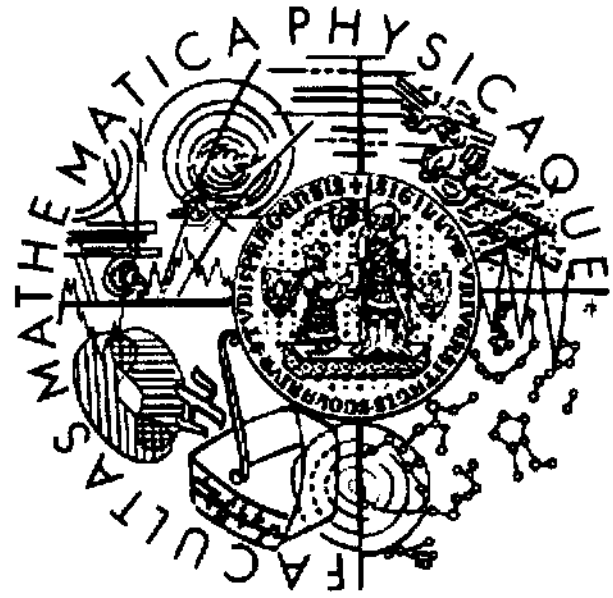
Charles University in Prague

Faculty of Mathematics and Physics

BACHELOR THESIS



Michal Žofka

Audio CD Ripper

Institute of Formal and Applied Linguistics

Supervisor

Mgr. Václav Klimeš

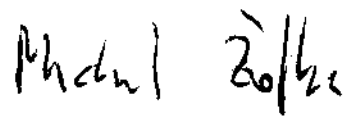
Study program: Computer science, Programming

2006

At this point I would like to give my thanks to Mgr. Václav Klimeš, the supervisor of my bachelor thesis, for giving me advice on my work as well as for pushing me to keep some deadlines as otherwise I would not have been able to finish the work in time.

Herewith I declare that I have written my bachelor thesis all by myself solely with the use of the quoted sources. I consent to the lending of the thesis.

Prague, August 11, 2006 Michal Žofka



Contents

Název práce: Audio CD Ripper

Autor: Michal Žofka

Katedra (ústav): Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Václav Klimeš

E-mail vedoucího: klimes@ufal.mff.cuni.cz

Abstrakt: Audio CD Ripper je jednoduchá aplikace pracující v prostředí WinXP, která možňuje digitální kopírování hudebních CD pomocí ASPI rozhraní či uložení jednotlivých stop CD nebo obrazu CD ve formátu WAVE na pevný disk. Po uložení lze spustit externí kompresor (LAME, APE nebo FLAC), ke kterému může uživatel specifikovat parametry příkazové řádky. Program se umí připojit jak k lokální, tak i ke vzdálené freedb databázi. Obsahuje editor ID3 tagů, funkce pro rychlé tagování, přejmenování souboru dle jeho tagu či vytvoření CUE sheetu vloženého CD. Uživatel si může zvolit vlastní schéma, podle kterého budou tvořena jména souborů. Audio CD Ripper lze použít též jako jednoduchý přehrávač CD.

Klíčová slova: freedb, DAE, ID3, ripper

Title: Audio CD Ripper

Author: Michal Žofka

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Václav Klimeš

Supervisor's e-mail address: klimes@ufal.mff.cuni.cz

Abstract: Audio CD Ripper is a simple application working under WinXP. It allows the user to digitally extract Audio CDs using the ASPI interface and save the tracks or CD image as WAVE files to the hard disk. It is able to call an external encoder (LAME, APE or FLAC) with user specified command line options to compress the files after extraction. Remote and local freedb connectivity is available. An ID3 editor is available as well as functions for automatic tagging and renaming files from tag. It is able to create a CUE sheet of the inserted audio CD. The user can choose the file naming scheme to be used for saving files. The application can also be used as a simple CD player.

Keywords: freedb, DAE, ID3, ripper

Chapter 1

Introduction

1.1 Ripping and Audio Rippers

Audio ripper is an application that has the ability to copy or extract the digital audio data stored on an audio CD and save it to the hard disk. This process is called ripping. Most audio rippers have other functions that can be useful when working with audio files.

1.2 Related Definitions and Explanation

Before getting started some explanation should be made first. This chapter contains a short introduction to the standards, file formats and other things closely connected with audio ripping.

1.2.1 The Red Book standard

Formerly, the compact disk format was defined for digital audio (CDDA). The Red Book standard was adopted from Sony and Philips by the Digital Audio Disc Committee and ratified as IEC 908. It defines and specifies the rules for CDDA. The standard is available in PDF format but unfortunately for the price of $210. Therefore I made use of the ECMA-130 standard for data interchange on read-only 120 mm optical data disks (CD-Rom) from June 1996. According to [11], it contains much of the information from IEC 908.

A small "file" called the Table of Contents (TOC) is at the beginning of a CD. The TOC is followed by strings of bits containing

the digital audio data (two channels of audio at 16 bits per channel and 44,100 samples per second). The separation of the audio data into individual tracks is done by the TOC. It contains pointers to individual tracks and tells the player where a track starts.

1.2.2 The WAVE file format

The Wave file format is a file format for storing digital audio data. As it is Windows' native file format the data values are stored in "Little-Endian" order. Some drives use the "Big-Endian" read order and a conversion is therefore necessary before storing the data. The format specifications are free available and can be obtained from various sources. One of these sources can be found at [12].

1.2.3 ASPI

ASPI (Advanced SCSI Programming Interface) was developed to provide a common language for communication between drivers and SCSI host adapters. It is used for communication with IDE devices and comes with the Windows installation. Some applications (e.g. Nero Burning Rom) install their own ASPI drivers.

1.2.4 Digital Audio Extraction (DAE)

When extracting through a sound card, the audio data is converted from digital to analog, resampled and converted back to digital. Because of this conversion the quality of the resulting audio file suffers. Digital Audio Extraction (DAE) does not use a sound card to extract audio data so there should be no noise or loss of fidelity in the resulting file. When an application sends a read request to the drive, it reads the digital audio data from the CD bit by bit and the application

receives the data exactly as it was read. Still some problems may occur while using DAE (see [8]):

- Jitter

  The cause of jittering is the inability of many drives to accurately seek a specific sector on an audio CD. When extracting, a block of sectors is read form the CD and written to the hard disk. Then the drive must seek the beginning of the next block of sectors. As the Red Book specification only requires the accuracy within $1/75^{th}$ of a second, the sector returned by the drive does not need to be exactly the one requested. Especially older drives tend to have jitter problems whereas newer ones (e.g. Plextor) perform jitter correction internally.

- Seek errors

  This goes hand in hand with the previous problem. Because of the Red Book inaccuracy the drive may return other track offset sectors even when extracting the same file. And there is no previous block to synchronize with as in jitter correction.

- Extraction speed

  High extraction speed may lead to seek errors during extraction on some drives because of the vibration. Only a few drives are able to extract at their full rated speed.

- Extraction offset

  Some drives have problems with extracting the first or last track on a CD because of having read the start and end times inaccurately by a few frames.

1.2.5 ID3 Tags

In addition to audio data a digital audio file can also contain additional data such as text and/or graphic. Most of the current audio players are able to extract and display this additional data. Writing the

additional data is called tagging. Eric Kemp developed the original standard for tagging digital files back in 1996. The term ID3 he used simply meant 'IDentify an MP3' at that time. ID3 tags were designed for the mp3 file format. There are two main versions: ID3v1 and ID3v2.

The ID3v1 is 128 bytes long and is appended to the end of the file. It includes title, artist, album, year, genre and a comment field. ID3v1.1 uses the last two bytes of the comment field to add the track number. As the tag is fixed-size, all the fields are space limited.

The ID3v2 tag is prepended to the file which means it is the first thing a player gets when playing the file. It is a chunk of data consisting of smaller chunks called frames. Every frame can hold up to 16MB of data and the whole ID3 tag can hold up to 256MB. These frames can contain any kind of information such as title, artist and many more. The biggest advantage of the ID3v2 tag (apart from almost no space limitations) is that it supports Unicode and is flexible and expandable. That makes adding new functions very easy. An ID3 tag parser simply ignores any frames that it does not recognize. The newest version is ID3v2.4.

For more information on ID3 tags visit [4].

1.3 The "planet's best audio ripper"

According to many sources and from my own experience Exact Audio Copy (EAC) is currently the best audio ripper around. The work on EAC has began already back in 1998. Since its 8 years of existence it has really developed and contains a whole lot of features. The audio extraction has become, as the name says, very exact. The CD has to be very scratched if EAC is not able to extract the tracks. EAC has a beginner mode that disables all advanced features and a Configuration Wizard. It allows inexperienced users to use it without any with any knowledge on audio ripping. It also includes an audio editor and CD writer.

But although I have been using it for 4 years now, half of the features not directly connected to ripping have never been useful to me. Some of the features are supererogatory (e.g. the audio editor) as I do not trust them and prefer to use an application that was developed especially for this use. And I have very much missed such simple features as quick tagging simply because they can save time. EAC does not allow editing ID3v2 tags and opens a dialog separately for every file that was chosen for editing even though some of the fields are the same (e.g. artist and album).

1.4 Audio CD Ripper

Because I am really interested in music and because I got fed up by applications that would not cover all the features I use when working with audio files, I decided to write my own application. It uses DAE as ripping audio CDs in analog mode through a soundcard has become really outdated. I wanted to create an application as good for extracting audio as EAC with additional features that would cover my needs. The first result of my effort is an application called Audio CD Ripper. Chapter 2.1 lists all features the application has so far. Although does not contain all the features I intended it to, I am quite satisfied with the result so far.

Version	**Features added**
0.1	fast reading mode, saving as wav files
0.2.1	Filename construction, CD player functionality
0.2.2	ID3v1
0.2.3	MAC support, editable track list
0.2.4	LAME and FLAC support, Saving options to registry
0.2.5	Editor for ID3v1 and ID3v2, renaming of files according to their Tag, create Tag from path
0.2.6	ripping CD Image, batch compression and decompression
0.2.7	Quick Tagging, CUE Sheet, setup, ASPI interface
0.3	Freedb connectivity, local freedb, CDplayer.ini, CD-TEXT, read commands and speed selection

Table 1.1: Version history of Audio CD Ripper

The version history of Audio CD Ripper is shown in table 1.1.

Chapter 2 contains the user documentation. It lists all the features and shows how to use them. The programmer documentation is in chapter 3. It describes only the most important functions, structures and classes. The topic turned out to be far more extensive than I had imagined. Which features were not added and what I intend to do is put down in chapter 4.

Chapter 2

User Documentation

2.1 Features

1. General
    - Usage of Windows XP Interface
    - CD player functionality
    - CD-Text support

2. Audio CD Extraction
    - Digital Audio Extraction (DAE)
    - Various read commands for different CD drives
    - Swapping left and right channels
    - "Big-Endian" byte mode
    - Extraction speed selection
    - Automatic creation of CUE sheets

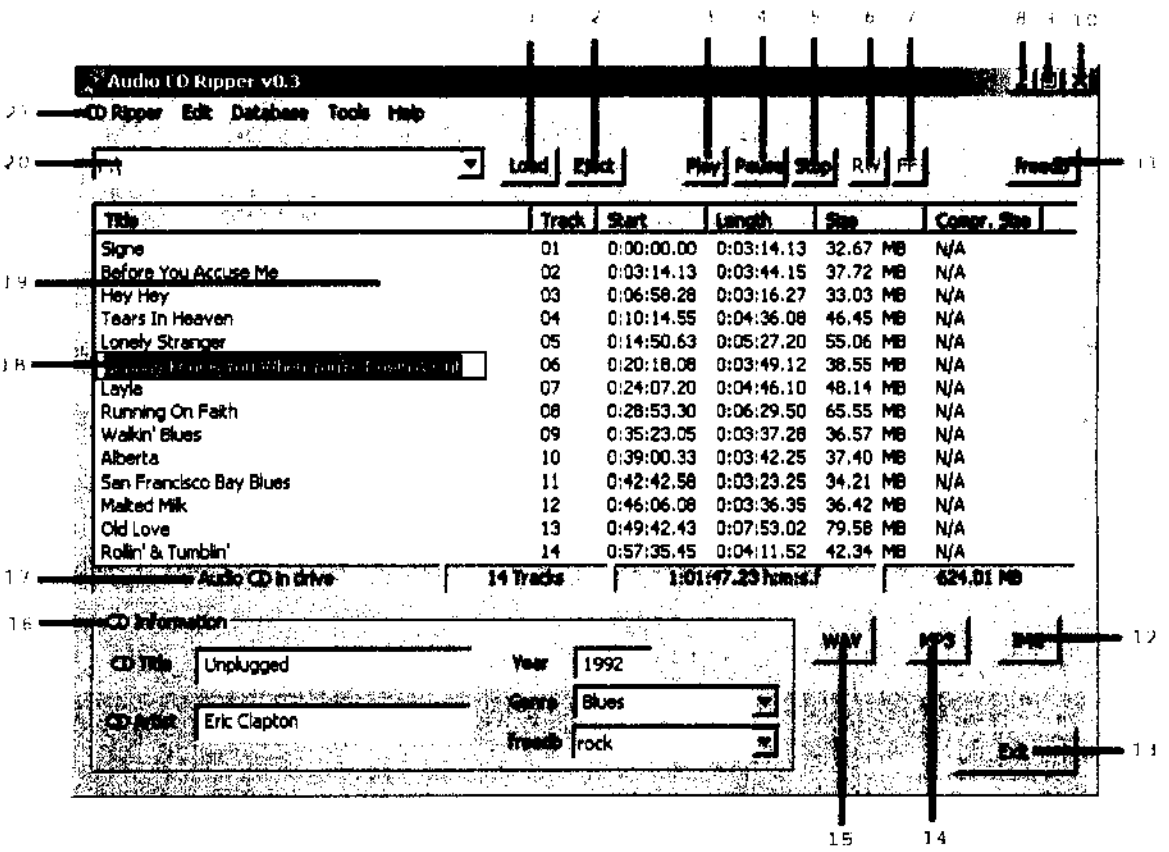
3. Compression
    - Batch compression and decompression of/to WAV files
    - Support of external encoders (MP3, APE, FLAC) for automatic compression after extraction

4. ID3 tags and freedb
    - ID3 Tag editor
    - Filename editing with local and remote freedb database and cdplayer.ini support

- Manage local freedb database (save, edit and delete CDs)
- Automatic rename of MP3 files according to their ID3 tag

2.2 Visual Overview

Screen shot 2.1 shows the main application window when a CD is inserted and its information is filled in. For a detailed description on the controls see the list bellow.

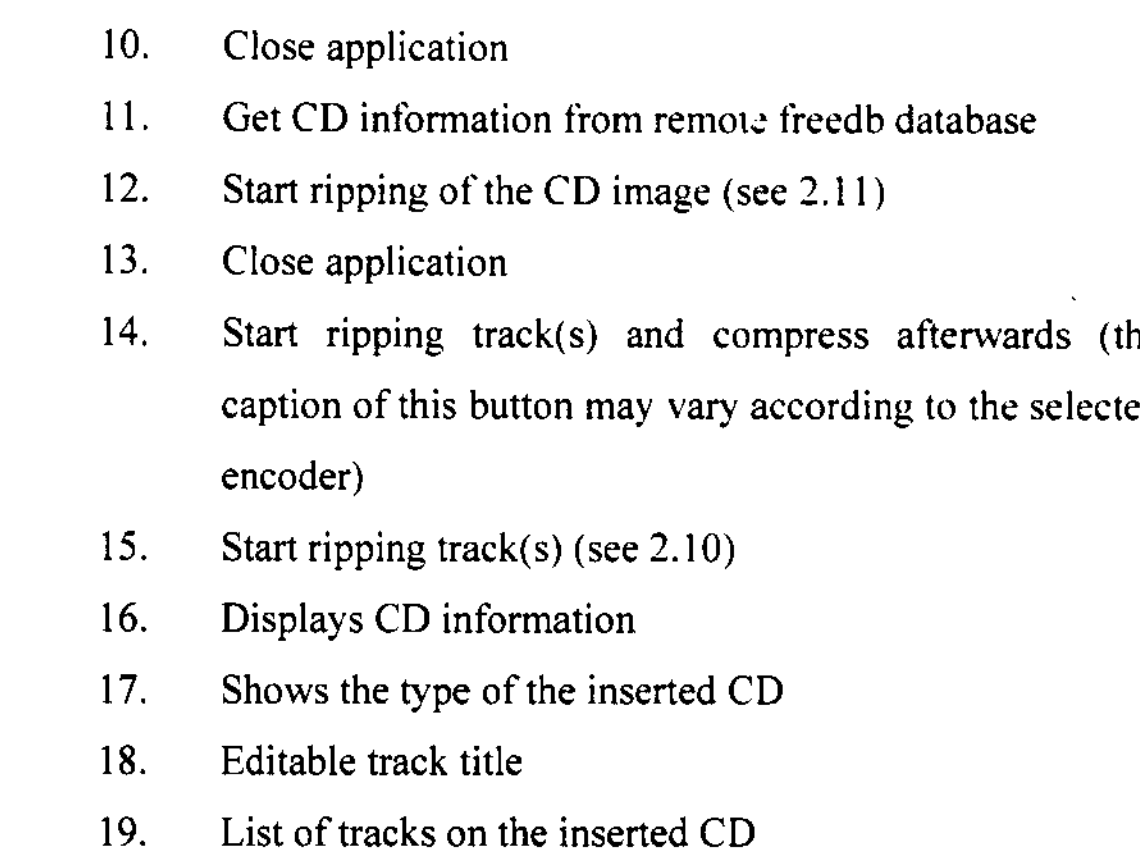


Screen shot 2.1: Main application window

1. Load drive tray
2. Eject drive tray
3. Start playback (focused track or first track) (see 2.8)
4. Pause playback
5. Stop playback
6. Move playback to previous song
7. Move playback to next song
8. Minimize window
9. Resize window (disabled)

10. Close application
11. Get CD information from remote freedb database
12. Start ripping of the CD image (see 2.11)
13. Close application
14. Start ripping track(s) and compress afterwards (the caption of this button may vary according to the selected encoder)
15. Start ripping track(s) (see 2.10)
16. Displays CD information
17. Shows the type of the inserted CD
18. Editable track title
19. List of tracks on the inserted CD
20. Selected CD/DVD drive
21. Menu

Screen shot 2.2 shows the CD information editor. It contains basically the same information as the main application window but shows it in the form it is submitted to the remote freedb database. For a detailed description on the controls see the list bellow.



Screen shot 2.2: CD information editor

1. Inserts the CD play order
2. Closes editor (if altered the CD information was altered it offers to save it to the local freedb database)
3. Shows how many times the information of the current CD was changed in the database
4. Fields in this column are editable and can contain user text
5. Clears all the information in the dialog (but does not delete the it from the local freedb database)
6. Closes editor (if altered the CD information was altered it offers to save it to the local freedb database)
7. Get CD information from remote freedb database
8. Submit CD information to remote freedb database
9. Save CD information to local freedb database
10. Delete CD information from local freedb database
11. Read CD-TEXT
12. Get CD information from local freedb database
13. Fields in this column are editable and can contain track titles

2.3 Installing the Application

To install Audio CD Ripper, run setup.exe and follow the instructions on your screen. You have to read and agree with the "End User License Agreement" to be able to install the application. By default the application folder is created in your Program Files folder. You can change the destination during the setup process. As a part of the application, flac.exe version 1.1.2, MAC.exe version 3.99 and lame.exe version 3.96.1 (with lame_enc.dll) are copied to the installation folder. These applications are used for external compression and are subject of copyright of their developers. For more information on these applications visit their web sites:

http://www.mp3dev.org/. http://www.monkeysaudio.com/, http://flac.sourceforge.net/ [5-7]. The application settings are written into the Windows registry on the first run of the application. Although the settings are set to the most common values you should configure the application before extracting any audio CDs (see chapter 2.5).

2.4 Uninstalling/repairing the Application

To uninstall or repair the application, run setup.exe or use "Add or Remove Programs" from the "Control Panel" menu. If you wish to remove the application settings from your Windows registry run the application and choose Delete Settings in the CD Ripper menu. Otherwise these settings are left in the registry after you uninstall the application. Note that the settings are different for every user on your PC and therefore every user has to delete his settings separately.

2.5 Configuring the Application

The application settings can be configured via items in the CD Ripper menu. "Audio CD Ripper Options" contain general application settings, "Drive Options" contain the settings of your drive, "Remote freedb Options" specify settings for remote freedb connectivity, "Local freedb Options" specify where and how to store CD information either downloaded from freedb or filled in manually, "Compression Options" specify whether to compress the extracted audio data and which compression to use. These settings can be deleted by choosing "Delete Settings".

2.5.1 Audio CD Ripper Options

1) Audio CD Ripper can automatically access the online freedb or open the CD information dialog if the "On unknown CDs" box is checked and the appropriate action chosen. "On unknown CDs"

means that the information on the current CD is not obtained in your local freedb.

2) To be prompted before overwriting any audio files check the "Ask before overwriting files" box otherwise the existing file with the same name will be overwritten without asking.

3) All files are named according to the entered "Naming scheme". Its maximum length is thirty characters. Apart from the listed switches (%N, %T, %A, %Y, %C, %F) you can use characters (' ', '(', ')', '_', '+', '-', '.') with a maximum of five between every switch to separate them. By default it is set to %N %T.

4) You can specify a standard directory to extract to or be prompted to choose one before every extraction.

5) You can choose the process priority. It can be one of the following: idle, normal, high, realtime, bellow/above normal. By default it is set to normal and it is recommended to leave this setting alone.

2.5.2 Drive Options

Before extracting any CDs you should choose the appropriate read command according to your drive. As there is no real drive standard most manufacturers have their own read commands for extracting data from a CD. A wrong chosen extraction command may cause problems on SCSI systems. If you do not know which extraction command your drive uses, take a look at the "User Reported Drive Features Database" consisting of submissions of EAC users. You can find it at http://users.pandora.be/satcp/eacoffsets01.htm. By default it is set to MMC. Very few drives will have exchanged stereo channels on extraction. The "Big-Endian" byte mode is often used on drives from Philips, Grunding, Plasmon, IMS, Kodak, Ricoh, HP and some others. Select this option only if the extracted wav file seems to consist of noise. Below, you can choose the drive speed. Selecting "max" means that Audio CD Ripper does not change the drive speed. The number of

sectors to get each read can be a value between 1 and 54. The more sectors can be obtained each read, the faster the extraction will be. Naturally, it also depends on the selected drive speed. If your drive is capable of reading CD-TEXT you can enable it. Some older drives do not support CD-TEXT. This can lead to errors when the Audio CD Ripper attempts to read it. Usually, only burned CDs contain CD-TEXT. It should not be present on an original CD.

2.5.3 Remote freedb Options

By default the freedb server is set to "freedb.freedb.org", the path to "/~freedb/freedb.cgi" and the port to 80. This specifies a random server. It is recommended to choose a server nearest to your region. The server can be changed by pressing the "Show all server list" button. Press the "Update" button to refresh the list and show active servers. Select the appropriate server and activate it by pressing the "Select" button. Timeout specifies how long the application waits for response from the server. If you are behind a proxy server, check the "Use proxy" box and fill in the required information. You have to enter your e-mail address to be able to submit CD information to the freedb database. An e-mail is sent to this address from freedb if your submission has been rejected.

2.5.4 Local freedb Options

In order to be able to use the local freedb you have to specify a folder where the local database will be saved. You can also choose the format and encoding of the database. The UNIX format uses 8-character filenames whereas the Windows format 6-character with the characters "to" in the middle. By default the format is set to Windows and the encoding to UTF-8 as it can handle all languages. Audio CD Ripper is also able to obtain information from the cdplayer.ini file

located in your Windows directory. Some rippers (e.g. Exact Audio Copy) have the ability to export their local databases to this file.

2.5.5 Compression Options

The extracted files are saved as WAVE (44.100Hz; 16 Bit; Stereo) files to your hard disk. Additionally, you can choose to compress the files with an external encoder. There are three encoders to choose from: two lossless ones (FLAC version 1.1.2 and APE version 3.99) and a lossy one (LAME version 3.96.1). In correspondence to the chosen compression the appropriate extension is selected. These files correspond to the most recent stable versions available at the date of release of this application. You can download newer versions of these files at the corresponding homepages of their developers (see bellow). Audio CD Ripper will be able to use them. You only have to copy the files to the External directory in the installation directory. For more information on the encoders see below or visit the homepages [5-7].

LAME homepage: http://www.mp3dev.org/

FLAC homepage: http://flac.sourceforge.net/

APE homepage: http://www.monkeysaudio.com/

Audio CD Ripper offers to automatically delete the extracted WAVE file after it has been compressed with an external encoder in case you do not want to keep it. If you are compressing the files to mp3, you can choose to tag the files with ID3v1 or ID3v2 or both.

Here is a short description of the most important switches the encoders have:

- **LAME**

  LAME works without any additional command line options but the files will be in very low quality. They are compressed to a joint stereo with fixed 128 kbps MP3 file. If you want the files to be of good quality, you should use

one of the following presets with variable bit rate as additional command line options:

- **--preset standard**

  According to LAME developers this is quite high in quality and the resulting bitrate should be in the 170-210kbps range.

- **--preset extreme**

  According to LAME developers this preset will provide slightly higher quality than the standard mode and the resulting bitrate should be in the 200-240kbps range.

- **--preset insane**

  According to LAME developers this preset is the highest preset quality available. It has a constant bitrate of 320kbps.

- **FLAC**

  FLAC works without any additional command line options as well. Running FLAC any options is equivalent to running it with the --compression-level-5 option. You can choose a level from --compression-level-0 (fastest compression) to --compression-level-8 (highest compression).

- **APE**

  APE does not work without additional options. Therefore the options are set to –c2000 (normal compression) if no additional options are specified. You can choose a level from –c1000 (fast compression) to –c5000 (insane compression).

The additional command line options are not being used when decompressing files.

2.5.6 Delete Settings

If you choose "Delete Options" the application settings will be replaced by default settings next time you run the application.

2.6 Tools

This is a set of tools that are only indirectly connected to the function of Audio CD Ripper but they can be useful. These tools allow you to modify audio files stored on your hard disk. You can compress and decompress files using the available external encoders, create and edit ID3 tags (ID3v1 as well as ID3v2) or rename files according to their tag. All the tools are based on the same way of usage. First you choose the function and then you select the files the function will be performed on.

2.6.1 Compress WAVs

This function allows you to compress WAVE files saved on your hard disk. Before you start compressing, check in Compression Options that you have selected the encoder you want to use. For information about the process follow the Status field in the displayed dialog and the information in the DOS window of the encoder.

2.6.2 Decompress

This function is independent on the selected compression. It can decompress MP3, APE and FLAC files to WAVE (44.100Hz; 16 Bit; Stereo) files. For information about the process look at the Status field in the displayed dialog.

2.6.3 Edit ID3 Tags

The ID3 Tag Editor is displayed for every selected file. It shows the MPEG information as well as the ID3 tag information it contains. The check boxes ID3v1 Tag and ID3v2 Tag indicate whether and which ID3 tag version the file contains. Both versions can coexist in one file. The tags can be edited, copied from ID3v1 tag to ID3v2 and contrary wise. Note that no information is written to the file until you press the "Update" button or the "Remove" button.

2.6.4 Rename from ID3 Tags

The selected file(s) is tagged according to the naming scheme entered in Audio CD Ripper Options. The file(s) is renamed from whichever ID3 tag version is present. If the file contains ID3v1 as well as ID3v2 tag, the file is renamed according to ID3v2 tag. If the file does not have a tag an error message is displayed.

2.6.5 Create ID3v2 Tag from Path

This is a very specific tool. The tag (only ID3v2) is created from the path of the file. Therefore the path and the file name have to be in a certain format. The folder structure has to be as follows: "..\Genre\Artist\ReleaseYear CDTitle\TrackNr Title.mp3".

2.6.6 Tag Dialog

This is a simple and fast way of creating id3tags. Again, the naming scheme is used. Only ID3 fields not specified in the naming scheme are enabled. The name and track no. fields are never enabled as they are specific for each file and they should be therefore present in the naming scheme and in the file name. The file name is parsed

according to the naming scheme and written along with the user specified ID3 fields to the file.

2.6.7 Create CUE Sheet

This function creates a CUE sheet for the inserted audio CD. The CUE sheet is used when burning the CD image (e.g. with Nero) and can also be used for splitting the CD image into separate tracks. If you are extracting the image of a CD the CUE sheet is created automatically and saved in the same directory as the CD image.

2.7 Database

When extracting tracks from a CD to your hard disk it is practical to replace the standard track names with the ones corresponding to the CD. If no information on the inserted CD is stored neither in the local database nor in the cdplayer.ini all tracks are named Track 01, Track 02, Track 03... and so on in ascending order and the reset of the information fields is left blank otherwise the CD information is displayed automatically. If the "On unknown CDs" option in Audio CD Ripper Options is enabled, the application will do it for you otherwise you will have to request the CD information manually. With correct filled in remote freedb options the application will lookup the CD in the online freedb database and you can choose to save it to the local freedb so that you do not have to look it up again next time you insert this CD. However this can take a few seconds and you have to be aware of the fact that not every CD is included in the online CD database or that the CD is include multiple in which case a dialog with downloaded results pops up and you can choose the most suitable one. Be also aware that the downloaded information may not include complete information about the CD and/or contain typing errors. In that case you should correct the errors and resubmit it to freedb. Do not change the freedb category as the freedb server would consider it a

different CD and the database would contain two entries for the same CD. An error message is displayed in case the CD was not found and you will have to enter the information manually.

To submit information to the freedb server you should stick to the following rules:

- Never submit information on burned CDs as they may have different track offsets to the ones on the original CD (e.g. when the 2 seconds gap was added or the TOA method was used for burning).
- When submitting information on a CD with various artists, the artist should be included in the track name (format "artist / track name") and the CD artist set to "Various".
- Artists' names should be written "first name last name" and NOT "last name, first name".

For more information on the freedb database visit their homepage [2].

2.7.1 Edit CD Information

This function opens the CD Information dialog for editing and viewing information about the inserted audio CD. The CD information can be obtained in various ways:

- If the CD drive is capable of reading CD-TEXT and the inserted CD contains CD-TEXT, it can be obtained by using the "Read CD-TEXT" button.
- If you are connected to the internet, the "Query freedb" button downloads the information about the CD from the freedb.
- If the queried CD has already been inserted before and its information has been saved to the local database, this information is automatically loaded from the local freedb database. It can be reloaded by pressing the "Read from local freedb" button.

- In case the CD information was not found in the local database Audio CD Ripper tries to find cdplayer.ini located in your Windows directory to see whether this file does not include the required information.
- The last possibility is to enter the information manually.

The CD information can be saved to the local freedb database, deleted from it and submitted to the freedb database. The playback order field can contain a comma-separated list of track numbers which are indexed from zero. This information is not used very often as the playback order on a CD is given.

After closing this dialog, the entered data are automatically displayed in the main window.

2.7.2 Clear Current CD Information

This function clears the currently displayed CD information but it does not remove it from the local freedb.

2.7.3 Get CD Information from

If the "On unknown CDs" option is not enabled or you want to reload the CD information use one of the following possibilities:

- Local freedb: If the queried CD has already been inserted before and its information has been saved to the local database, this information is loaded from the local freedb database.
- Remote freedb: If you are connected to the internet the information about the CD is downloaded from the freedb database. This function is equivalent to the function of the "freedb" button in the main window.
- CD-TEXT: The CD drive has to be capable of reading CD-TEXT and the inserted CD has to contain CD-TEXT.

2.8 Playing Audio CDs

Audio CD Ripper can be used as a simple CD player. It uses the MCI interface for playback. Press the "Play" button to start the playback. It starts with the selected track or at the beginning of the CD if no track is selected. The other buttons do exactly what you suppose they would (RW jumps a track back and FF a track forward).

2.9 Manipulating the Track List

This works in a similar way as in Windows Explorer. Thus use the Shift and Control keys to select multiple tracks. You will find two helpful functions in the Edit menu: Select All and Invert Selection. Click on a track to select it. If you want to rename a track, select the track and choose either Edit → Rename Track or click on the track with the left mouse button. After you have entered the track title press Enter to move to the next track which will be automatically in an editable state or press ESC to cancel further editing.

2.10 Extracting Tracks

Select the tracks you want to extract. If you have not selected any tracks Audio CD Ripper will assume that you want all tracks on the CD to be extracted and selects them for you. To save the files in uncompressed form press the "WAV" button, to save them in compressed form press the "MP3/APE/FLAC" button (its caption depends on the selected external encoder). Choose the directory the files will be copied to in the displayed Save dialog (no dialog is shown when the option "Use this directory" is enabled). If you enter a filename it will be ignored as Audio CD Ripper creates the filenames according to the naming scheme. The extraction process is displayed in the Process dialog. You can follow the extraction process of every track. The filename specifies which track is currently being extracted.

The status of every extracted track is shown in the Status field. You can choose to cancel the currently extracted track by pressing the "Skip File" button or cancel the whole extraction by pressing "Cancel". When the extraction is finished, press the "OK" button to close the progress dialog.

2.11 Creating CD Image and CUE Sheet

Creating the image of an Audio CD can be useful in many ways. You can store compressed images of your CDs or leave the image uncompressed to burn it. The image can be created by pressing the "IMG" button. Choose the directory the files will be copied to in the displayed Save dialog and enter the filename. If you do not enter one the file is saved as CDImage.wav. If a compression is selected the file will be compressed after extraction otherwise it is left uncompressed. The extraction process is displayed in the Process dialog. The status of the extracted image is shown in the Status field. To cancel the extraction press "Cancel". When the extraction is finished, press the "OK" button to close the progress dialog. The CUE sheet is created automatically and saved in the same directory as the image file.

Chapter 3

Programmer Documentation

This chapter analyzes the individual units that form this application. Audio CD Ripper uses id3lib.lib and id3lib.dll for writing ID3 tags and retrieving MPEG header information. The ASPI interface uses the wnaspi32.dll if it is present (either the one from the System directory or the one installed by NERO).

3.1 CDRipper

This is the program's main unit. It handles almost all dialog functions, works with the Windows registry and runs the main functions in other units. Before the application is started it checks whether it is really running under WinXP. It loads the default settings and *readini()* tries to query the Windows registry for the user specified settings. When the application is closed *writeini()* writes the settings to the Windows registry unless the user chose to delete these settings. In that case *deleteini()* is run instead.

The functions to display messages may seem a bit odd. They are prepared for the application to be multilingual. The first argument given to the function is a number specifying the message in the selected language file.

RenameFromTag() renames a file according to its tag. If both versions are available it chooses ID3v2. The function cuts the filename from the path and replaces it with a new filename which is created using the tag fields while applying the selected naming scheme.

NamingStr() creates the input string a *scanf()* function can use to parse a string. *ParseNamingScheme()* uses this string to parse the filename it receives into ID3 tag fields.

3.2 Mp3Info

Mp3Info class was written to deal with ID3 tags. It is defined in the source code snippet 3.1.

```
class Mp3Info
{
public:
  inline Mp3Info(void):// Left out
  {  }

  virtual ~Mp3Info(void)
  {     Free();   }
  bool Init(const char* cszFilename);
  bool Update(int ulFlag, bool tagVersion);
  void TagAfterCompress(int i);
  void ReAlloc(HWND hDlg, bool tagVersion);
  bool ParsePath(char* path);
  void Clear(flags_t tag);
  void FreeStrings(void);
  void FreeString(char *str);
  void Free(void);

  int szError;
  /* File info */
  char* szFilename;
  bool  bHasLyrics, bHasV1Tag, bHasV2Tag;
  int   nFileSize;
  /* Header info */
  int nCbrBitRate, nVbrBitRate, nBitRate, nSampleRate, nLength;
  int nFrames;
  char *szBitRate, *szMpegLayer, *szMpegVersion, *szChannelMode;
  char *szCRC, *szCopyrighted, *szEmphasis, *szOriginal;

  char* tmpStr;
  /* V1 up */
  char *szArtist1, *szTitle1, *szAlbum1, *szComment1, *szTrack1;
  char *szYear1, *nGenre1;
  int nTrack1, nYear1, nGenre1;
  /* V2 only */
  Char *szArtist, *szTitle, *szAlbum, *szComment, *szTrack, *szYear;
  char *szComposer, *szCopyright, *szEncodedBy, *szOriginalArtist;
  char *szURL, *szGenre;
  int nTrack, nYear, nGenre;

protected:
  ID3_Tag* m_id3tag;
  const Mp3_Headerinfo* m_mp3header;

  char *ID3_GetV2Frame(ID3_FrameID frameID, ID3_FieldID fieldID);
  ID3_Frame* ID3_AddV2Frame(ID3_Tag *tag, const char *text,
           bool replace, ID3_FrameID frameID, ID3_FieldID fieldID);
  size_t ID3_RemoveV2Frame(ID3_Tag *tag, ID3_FrameID frameID);

  void ID3_ParseHeader(void);
}
```

Source code snippet 3.1: class Mp3Info

The definition of the constructor is left out as it does nothing but initializing all class members. Not only does the class deal with ID3 tags, it reads and parses the MPEG header of the file as well. ID3v1 and ID3v2 are completely separated as it allows both versions to hold different data which may sometimes be useful especially because the ID3v1 is space limited. All members whose names end with "1" belong to ID3v1. Members belonging to ID3v2 do not have this special end character. *Init()* tries to read the ID3 tags and the file header. *Update()* writes the new tag(s) whereas *Clear()* removes the present tag(s). *TagAfterCompress()* retrieves information from *TcdInfo* (see 3.5) and copies it to the corresponding class members of *Mp3Info*. *ParsePath()* parses the path of the tagged file and writes it to the file unless there has been an error while parsing the file path (e.g. the file path did not have the right structure). The *Update()* function is not needed in this case.

3.3 Compressor

This unit is not very interesting. It simply creates the switches for the given external encoder, the input filename, output filename and the user specified additional command line options, and creates the appropriate thread. FLAC and LAME both do not need any switches to be specified for encoding. When FLAC encodes with no switches specified it behaves as if it was given the switch "-5". LAME with no switches encodes to a joint stereo with 128kbps and low quality. To decode a file FLAC needs the "-d" switch which the user does not have to specify, it is added automatically. It is the same with LAME except that the switch is "--decode". APE is the only one that requires a switch to be specified. Therefore I decided to set the switch to "-c2000" (normal compression) if no user specified switches are available. The switch for decoding ("-d") is added automatically as well.

3.4 ASPI

The usage of an ASPI interface has many advantages such as drive speed selection or read command specification. Source code snippet 3.2 shows the enumeration type of the implemented read commands.

```
enum READMETHOD {
        READMMC, READMMC2, READMMC3, READMMC4,
        READ10, READNEC, READSONY,
        READC1, READC2, READC3
};
```

Source code snippet 3.2: enum READMETHOD

For more information about the MMC commands take a look at T10, the homepage of the "Technical Committee of the InterNational Committee on Information Technology Standards (INCITS)" [4]. The first-generation command set for multi-media devices based upon SCSI-2 is described in MMC. The second-generation command set (MMC-2) for multi-media devices includes initial support for DVD. The third-generation command set (MMC-3) for multi-media devices includes -RAM, -R, -RW, and +RW and introduces CD-MRW and DDCD. The fourth-generation command set (MMC-4) for multi-media devices has the next major inclusion of support for DVD+R, +MRW. I was not able to find the specifications on the other read commands.

InitASPI() tries to load the wnaspi32.dll from the system folder. If it fails it tries to find the wnaspi32.dll installed by Nero. The return value specifies whether the loading was successful or not. If it was not successful the *SendASPI32Command()* that should have been loaded is replaced by *NtScsiSendASPI32Command()* which converts the ASPI-style SRB to SCSI pass through IOCTL.

One of the main functions in this unit is *sendASPI()*. It sets the arguments for *SendASPI32Command()*, adds the command and calls this function.

Rip() creates the command according to the chosen read command and extracts the given number of sectors from the CD to a buffer. If

the channel swapping or the "Big-Endien" byte mode are enabled it does the changes to the buffer and returns the buffer to the *ripThreadFunc()*.

RipThreadFunc() extracts the audio from the CD. It extracts one track or the whole CD image according to the arguments it was given. First it gets the offset of the track according to the TOC, sets up the WAVE file header and creates the file. It calls the *getSectors()* and writes the buffer filled by this function until there are no sectors left to read. If *getSectors()* or writing the buffer to the file fail, *ripThreadFunc()* is aborted and the file closed and deleted.

3.5 CDMusic

This unit handles the Edit CD Information dialog and the dialog with multiple results from local or remote freedb. The basic element in this unit is the structure *TcdInfo*. As the name says it deals with information about the CD. That means it queries local and remote database and stores the information in the structure members. It uses the ASPI interface to send ASPI commands such as reading the TOC of a CD, reading CD-TEXT, reading sectors from the CD or setting drive speed. Its definition is shown in the source code snippet 3.3. Some of these functions are defined in ASPI.cpp or freedb.cpp but for complexity they will be explained here.

It is obvious from the definition that the functions *getTOC()*, *getCDText()*, *getSectors()* and *setSpeed()* do nothing apart from querying the ASPI interface through the function *aspiCommand()* with a different parameter. The *aspiCommand()* function tries to use *aspiCmd()* first and *ntscsiCmd()* if the previous failed. If both functions fail it sleeps for 20ms and tries again. It quits trying after both functions have failed five times. *AspiCmd()* tries to load the functions *GetASPI32SupportInfo()* and *SendASPICommand()* from wnaspi32.dll. In case it does not succeed *ntscsiCmd()* comes in the game and it replaces *SendASPICommand()* with

```
struct TcdInfo
{
  HINSTANCE hInst;
  int trackN2;      //number of tracks
  DWORD total;      //disk length in seconds
  int revision;     //how many times has been freedb record modified.
  TCHAR
*Artist,*Album,*Year,*Genre,*Extd,*Order,*Category,*SubmittedVia;
  TCHAR *Dtitle;     //used during parsing the file
  TCHAR *inetCategory; //category read from freedb server
  TCHAR *inetDiscId;   //discId read from freedb server
  int inetRevision;    //revision read from freedb server
  TCHAR discId[12]; //freedb identifier
  TCHAR mediaIdHex[16];

  Darray<TmultipeResultInfo> results;
  bool isExact;
  bool modif;

  Ttrack Tracks[Mtrack];

  int aspiCommand(int action);
  int getTOC(){ return aspiCommand(0); }
  int getCDText(){ return aspiCommand(5); }
  int getSectors(){ return aspiCommand(1); }
  int setSpeed(){ return aspiCommand(2); }
  void findInfo();
  void reset();
  void resetResult();
  void resetFields();
  void readCdplayerIni();
  void readLocalCDDB(TCHAR *mask=0);
  void saveToLocalCDDB();
  int deleteFromLocalCDDB();
  void cddbQuery();
  int getFromFreedb(bool save);
  void submit();
  void computeDiscID();
  TCHAR *getArtistAlbum();
protected:
  void parseLine(char *key, char *value);
  void parse(int(*get)(void*),void *param, TCHAR *cat);
  void parse(FILE *f, TCHAR *cat);
  void openCDDBfile(TCHAR *fn, TCHAR *cat);
  void rdFromString(char *s, TCHAR *cat);
  void wrToString(Darray<char> &s);
  TCHAR *getCategoryFolder(TCHAR *fn);
  void addResult(char *&line);
  void saveToLocalCDDB(char *content);
  int driveCmd(int action);
  int aspiCmd(int action);
  int ntscsiCmd(int action);
  int parseCDText(BYTE *buf);

  int enc;
  int trackOffset;
};
```

Source code snippet 3.3: struct TcdInfo

NtScsiSendASPI32Command() which converts the ASPI-style SRB to SCSI pass through IOCTL. *SendASPICommand()* is being used by *sendASPI()*. Apart from that, *aspiCmd()* and *ntscsiCmd()* just call *driveCmd()*. This function is finally the one that makes use of the parameter that was given to *aspiCommand()*.

The *findInfo()* function retrieves the CD information. First it tries to retrieve it from the local freedb. If the information is not there it tries to find it in cdplayer.ini. If cdplayer.ini does not exist or contain the required information it searches for CD-TEXT. And finally, if all these efforts have failed, it connects to the remote freedb.

The functions *readLocalCDDB()*, *saveToLocalCDDB()* and *deleteFromLocalCDDB()* take care of the local database. If the UNIX format is selected every CD entry is saved to a different file. The file name corresponds to the disk ID. The Windows format is a little bit different. It is only 6 characters long and the two characters in the middle are set to "to". The first two and last two characters specify the range of disk IDs that are saved in the given file. *ComputeDiscID()* is responsible for determining the disk ID. *SaveToLocalCDDB()* saves the current CD information. *WrToString()* creates the string that will be stored in the database. If the Windows format is set the new entries are prepended the file. If the file size exceeds the limit it is split with *splitFile()* into two files. *SplitFile()* reads the whole file into the memory, extracts all disk IDs and sorts them. After the IDs are sorted the median is determined and the entries are split according to it into two files. *ReadLocalCDDB()* looks through the whole database for the given disk ID as it can be stored in various categories and returns all corresponding results. *Parse()* is used for parsing the CD information read from local freedb database. *ParseLine()* parses every line of the information and saves it to the appropriate *TcdInfo* members. The *getArtistAlbum()* is only used when displaying multiple results. The information shown in the multiple results dialog is freedb genre, disk Id and Artist/Album. *DeleteFromDatabase()* does not delete all the entries with the given disk ID but only the currently edited one.

The following functions deal with the remote freedb. *getFromFreedb()* gets the result(s) from the freedb server using *cddbQuery()*. The result string returned by the freedb server is parsed by *rdFromString()*. The parameter given to *getFromFreedb()* tells the application whether to display the selected result automatically to the

local database or to display it only. *Submit()* sends the CD information to the remote freedb. See chapter 3.7 for more information on freedb.

3.6 WAVE (and RIFF)

Not all the functions from the *WaveFile* class are used as the application only writes a WAVE file but never has to read it. The constructor does nothing apart from initializing the class members. *SetupFormat()* sets up the WAVE file header values according to the arguments it was given. The arguments are: sample rate, bits per channel and the number of channels. Additionally it sets the values of byte per second and byte per sample. *OpenWrite()* creates the file, opens it for writing and calls *WriteHeaderToFile()* to write the header of the file in the form it was defined by *SetupFormat()*. Then it positions the file pointer to the appropriate point to start writing sample data. The result is a "RIFF WAVE" file. *WriteBuffer()* tries to write the buffer, it was given a pointer to, to the file. If the number of bytes the buffer contains differs from the bytes that were written to the file an error is returned. *Close()* looks whether a change was made to the file and writes the appropriate changes to the file's header (e.g. the whole length of the file and the length of the audio data) before closing the file.

3.7 freedb

Before being able to access the remote freedb database (and the local as well) the disk ID has to be computed. It is an eight-digit hexadecimal (base-sixteen) number, computed using data from a CD's Table-of-Contents (TOC) in MSF (Minute Second Frame) form. To compute the disk ID of a CD Extra the data track has to be included. If the disk ID is less than 8 characters long the matching number of zeroes is prepended. For more detailed information on the disk ID

computing and the algorithm, take a look at the DISCID Howto article in the freedb FAQ [2].

There are two methods of submission to the freedb database: by e-mail or HTTP. The application uses the second method. The data is transmitted to the database through a CGI program, in this case the submit.cgi. The address for submissions is http://freedb_server/~cddb/submit.cgi where freed_server is replaced by the freedb server selected in remote freedb options. The "POST" method has to be used for sending the data. The *submit()* function checks whether the user has entered his e-mail. If the e-mail address is not specified an error message box is shown to the user telling him where to enter it. Then it checks if ale required CD information was entered by the user. The artist, album, freedb category and track titles have to be entered. The user will not be permitted to submit until he has filled in all the required fields. Before submitting the CD information to the server the function checks with the server whether there is no entry with the disk ID already in the given category. If it is not present, it prepares the http header appends the data string and calls *command1()* (will be discussed later in this chapter).

There are also two methods for accessing the freedb server: CDDBP and HTTP. As in the previous case the application uses the second one. All current freedb servers answer at IP port 80 (for HTTP access). The server address for retrieving is http://freedb_server/~cddb/cddb.cgi where freedb_server is again replaced by the current user selected one. *Command()* prepares the handshake command string which is made via the "cddb hello" command and calls *command1()*.

The transmission is obtained by *command1()*. Firstly, it calls *startConnection()*. This function initializes the WinSock and returns whether it was successful. The GET or POST request is sent via *wr()* and the response is received via *rd()*. The socket can be disconnected thereafter.

For more information on the CDDB-protocol, database-format specification, local and remote access and submission to the freedb server, visit [2].

Chapter 4

Conclusion

4.1 The Result

The result of my work is quite pleasant. Audio CD Ripper has ten different read commands for different drives. It can swap audio channels and low and high bytes before writing the ripped audio data to a file. Setting the drive speed or specifying the number of CD sectors to get each read can be useful as well but it is mainly the preparation for error corrections such as speed reduction during extraction. The access to local and remote freedb database saves a lot of time and typing. The database has the same structure as the one on the freedb server whereas EAC has its own structure saving the information to one file. New etries are appended to the database file. That gives Audio CD Ripper a few milliseconds advantage when reading the information from the local freedb.

What I had added especially for myself, is the function tagging mp3 files from path. It saves me a lot of time as EAC only allows ID3v1 to be edited and one has to write the same information again and again. For people using a slightly different directory structure than me I added the other Quick Tagging function. There is no need to write any information more than once. The rest is done automatically. It supposes of course that the mp3 files have some relation, e.g. they are from the same CD.

What I learned is the fact that it is not just the application that rips the audio; it is also the CD drive the rip is made with. A good CD drive can do a lot of work. For example most of the Plextor CD drives

do jitter correction themselves. One of the useful functions of EAC I did not have time to add yet is comparing two audio files against each other for differences. I tried to extract several CDs with Audio CD Ripper and with EAC and compared them against each other. (I used a Plextor PX-750A, an ASUS DRW-1608p3S and a notebook drive.) Files ripped from an unscratched CD with Audio CD Ripper and EAC on the Plextor drive compared to each other were identical. Files ripped on the ASUS drive were identical in most cases whereas all files ripped on the notebook drive showed a lot of different samples when compared to each other. I must admit I did not hear anything out of place when I listened to the files ripped with Audio CD Ripper but EAC is naturally much better than my hearing ever will be.

4.2 The Next Step

The initial idea was to create an application as good for ripping audio as Exact Audio Copy (EAC) omitting some features for work with audio files and adding a few ones I missed when using EAC. The reasons why I was not able to add all the features and ideas I had in mind are quite clear. The first one was lack of time. EAC is being developed for eight years. I only had one and a half years. The second reason was that I had very little or none experience with the given topic. And the last reason was that I had never before written a project of this size.

The first thing that I will have to do is to remove the currently biggest weakness of Audio CD Ripper: its error correction features. Reading every sector twice will be added for a start. Other features like using the C2 error information, speed reduction on errors or jitter correction will follow soon after. There is still a lot of work to do. The current state of Audio CD Ripper is a good mile stone for future work. I can continue with what I am interested in and probably take the application to a whole new level in my diploma thesis.

Literature

[1] Andre Wiethoff's Exact Audio Copy homepage http://exactaudiocopy.de

[2] freedb homepage, http://freedb.org/

[3] Technical Committee T10 homepage, http://t10.org/

[4] ID3v2 homepage, http://www.id3.org/

[5] FLAC homepage, http://flac.sourceforge.net/

[6] Monkey's Audio homepage, http://www.monkeysaudio.com/

[7] LAME project homepage, http://www.mp3dev.org/

[8] Fries, Bruce; Fries, Marty: The MP3 and Internet Audio Handbook, TeamCom Books, 2000, http://www.teamcombooks.com/mp3handbook/15.htm

[9] Petzold, Charles: Programming Windows, Microsoft Press, 1998

[10] Standard ECMA-130, second edition, 1996, http://www.ecma-international.org/publications/standards/Ecma-130.htm

[11] Chapin, Chip: CD-DA (Digital Audio) 1, Chip's CD, 2005 http://www.chipchapin.com/CDMedia/cdda1.php3

[12] The Sonic Spot, Wave File Format, http://www.sonicspot.com/guide/wavefiles.html