Abstract

Due to the ever increasing number of digital audio files available (through legal and not so legal download sights) to people, many people have a large number of files which are hard to search through. The project is to construct a GUI application to view edit and update extended information stored within digital audio files, and to sort thease files into directories based on the extended information.

This report first looks into the problem in more detail then goes to cover the background material including libraries used and technical information on relevent technologies. It then follows through the developmeant of the programme including areas of difficulty and problems encountered. The final part of the report evaluates the programme at the end of the aloted time.

Acknowledgements

I would like to thank The staff at the University of Heartfordshire for their help and support especialy R. Dickerson (Bob) who has helpd a lot with guidance for this project.

I would also like to thank My family and friends who have supported me with testing and feedback on the project throught the year.

Contents

Chapter 1

Introduction

More people around the world are using electronically stored music for a number of reasons including ease of use, portability and ease of sharing/distributing.

The most popular format of digital music people are storing on their computers is MP3. These files (like many other digital audio files) can store additional information about the recorded track including artist and track names. However these aren't always entered properly and if someone has a large number of mp3 files in one directory it is hard to locate the desired file. In adition to this file names are also often misrepresenting or not formatted in a manar that people would prefer.

Due to the nature of most computer users I wish to creait a GUI (graphical User Interface) application as manu users are unfamilier/unwilling to use CLI (Command Line Interface) applications. Annother more personal gole for this project (that does have a lot of implications) is to construct a platform independent application (can be used on Windows, Mac and Unix systems) due to the growing popularity of Linux.

The project is to construct a platform independent GUI application to view edit and update extended information stored within digital audio files, and to allow a user to sort thease files (into directories) based on the extended information.

Chapter 2

Background

Here i have documented a number of areas including libraries and file layouts that i have had to research and understand as part of my project. I have structured this section in a non chronalogical order and have attempted to organise items by relevence to eachother. First of all i have the libraries that i have looked into. Then i've got all the information on digital audio files (and related aspects). A small number of miscelanious items then finaly i have looked into some existing applications similar or related to my project.

wxWidgets (wxWindows)

This library (formerly known as wxWindows) works with C++ and python however it can be expanded to other programming languages. It has been under development for 11 years with a number of GNU and commercial programmes being developed with it. One program that I looked at is "audacity" which is a platform independent audio editing application (i tried it both under Linux and Windows.

2.1 WxWidgets(WxWindows)

2.1.1 Background

from [Smart, 2005].

> wxWidgets is a set of libraries that allows C++ applications to compile and run on several different types of computer, with minimal source code changes. There is one library per supported GUI (such as Windows, GTK+, Motif, and Mac). As well as providing a common API (Application Programming Interface) for GUI functionality, it provides functionality for accessing some commonly-used operating system facilities, from copying and deleting files to socket and

thread support. wxWidgets is a 'framework' in the sense that it provides a lot of built-in functionality, which the application can use or replace as required, thus saving a great deal of coding effort. Basic data structures such as strings, arrays, linked lists and hash tables are also supported.

2.1.2 Features

Here is a list of features of the wxWidgets library showing the broard range of aspects supported by the library, It shows how most basic features of operating systems are catered for making this library ideal to use even possably in no GUI applications.

basic GUI

The library allows you to quickly construct simple GUI applications starting with a basic window which can then have widgets added like the ones documented below:

Text Boxes These are highly configurable they can be simple one line "text boxes" but are also powerful enough to become a basic editor for text or RTF files.

List Boxes A widget that allows for a number of items to be selected (then actions can be performed with the selected information). There are actualy two different types of list box with a variaty of different properties.

Drop Down Boxes These allow for a number of items to be selected from a "drop down" list. similar to a one line text box.

Image Boxes Thease contain images.

tick Boxes Basic tick box (there are a number of similar widgets for boolean selection)

Menu Bars Drop Down (and pop up) menus for a GUI application.

Basic Data Types

Due to the differences between data types from one platform to annother (such as handeling of strings) WxWidgets has its own variaty of basic data types including the ones listed below.

WxChar This is similar to the normal char in C++ but provides behaural clarity across platforms.

WxString This is a String type that has a wide variaty of functions that can be performed on it including pattern matching and convertion functions.

WxPoint This is a datatype containing two intagers to plot a position on the screen.

WxSize This is a datatype similar to the WxPoint except rather than holding a poinbt it holds a size (in pixels).

File procedures

Within the library are a large number of file Input/Output Functions to create, copy, rename, delete, read and write to files and directories. Mainly because the library is aimed at platform independence and all file functions are heavily platform dependent, this makes it very easy to construct a platform independent application without having to write seperate windows, linux, mac vertions of the File I/O functions.

Network Features

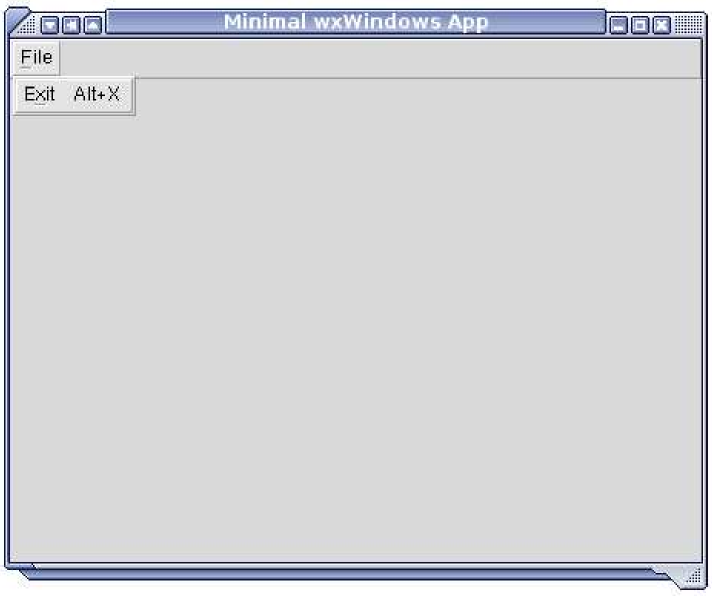
There are a small number of network functions built into the library. Thease again are to aide in the platform independence of applications developed with the wxwidgets librarry. The network functions cover all the lower level networking features of an application but don't contain all the features of a specialist network library that may include more advanced protocol integration.

2.1.3 Sample minimal Application

Here I have included a screenshot of a minimal application in wxWidgets and below the code (and a code description).

Screen Shot of minimal application

This is the simplest application you can creait with the library that creaits a window with a drop down menu.



Sample Code

Below I have included (and documented) some sample code to demonstrate the use of the wxWidgets library.

```
01 #include"wx/wxprec.h"
02 #include"wx/wx.h"
03 class MyApp : public wxApp
04 {
05 public:
06     virtual bool OnInit();
07 };
08 class MyFrame : public wxFrame
09 {
10 public:
11     MyFrame(const wxString& title, const wxPoint& pos, const wxSize& siz
12             long style = wxDEFAULT_FRAME_STYLE);
13     void OnQuit(wxCommandEvent& event);
14     void OnAbout(wxCommandEvent& event);
15 private:
16     DECLARE_EVENT_TABLE()
17 };
18 enum
19 {
20     Minimal_Quit = 1,
21     Minimal_About = wxID_ABOUT
```

```
22 };
23 BEGIN_EVENT_TABLE(MyFrame, wxFrame)
24     EVT_MENU(Minimal_Quit,  MyFrame::OnQuit)
25     EVT_MENU(Minimal_About, MyFrame::OnAbout)
26 END_EVENT_TABLE()
27
28 IMPLEMENT_APP(MyApp)
29
30 bool MyApp::OnInit()
31 {
32     MyFrame *frame = new MyFrame(_T("Minimal wxWindows App"),
33                                  wxPoint(50, 50), wxSize(450, 340));
34     frame->Show(TRUE);
35     return TRUE;
36 }
37
38 MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize
39        : wxFrame(NULL, -1, title, pos, size, style)
40 {
41     wxMenu *menuFile = new wxMenu;
42     menuFile->Append(Minimal_Quit, _T("E&xit\tAlt-X"), _T("Quit this pro
43     wxMenuBar *menuBar = new wxMenuBar();
44     menuBar->Append(menuFile, _T("&File"));
45     SetMenuBar(menuBar);
46 }
47
48 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
49 {
50     Close(TRUE);
51 }
```

01 Include This includes the wxWidgets precompilation library.

02 Include This includes the main library file (there are a number of others) for the wxWidget library.

03 - 07 Class MyApp This creaits a public class that is used to conect to the library. this class is called to start the GUI part of the application later on.

08 - 17 Declaration of MyFrame. This Defines the functions and variables of the main frame in the GUI.

18 - 22 enumerations used to give numeric values to events in the GUI (such as clicking on a button).

23 - 26 Event table this links the enumerations to functions that they call.

28 This calls the inherited functions to start up the application (this in efect contains the “main” function.

30 - 36 OnInit() The OnInit function is called when the application starts

32 This creaits a frame (of type MyFrame) giving it a title(”Minimal wxWindows App”), position on the screen ‘wxPoint(50, 50)’ and a size ‘wxSize(450, 340)’.

34 show The frame-¿show command makes the application display the frame that has just been creaited.

38 - 46 constructor for MyFrame. This creaits the frame and also adds to the frame a menu.

48 - 51 This is the function called when the application is exited, this allows for the program to have a number of functions or commands called when it exits including an “Are you sure you want to exit” message.

2.2 Lua

Lua means moon in Portuguese and is pronounced LOO-ah[Lua-website, 2005].

> Lua is a powerful light-weight programming language designed for extending applications. Lua is also frequently used as a general-purpose, stand-alone language. Lua is free software.

Lua is designed and implemented by a team at Tecgraf, the Computer Graphics Technology Group of PUC-Rio (the Pontifical Catholic University of Rio de Janeiro in Brazil). [Lua-website, 2005]

2.2.1 Description of language

Lua is designed as an interpreted language with dynamic type checking. However there is Luac which is a compiler that does the type checking but does not produce a binary file that can be executed, the file still needs to be used with the Lua program.

The largest common use of Lua is building it into other programmes to extend them. There is a powerfull C API that allows programmes to use Lua for scripting in larger applications. Lua has a large following in the Games development industry for its scripting uses. It can also be used to read INI like config files as Lua program files that specify assignmeant of values to objects look like INI files. A text file can be created and passed by Lua with the values passed to the application, this means that an application developer does not have to write a full in depth class for input from an ASCII “options” file (this is the functionality I’m most interested in.

2.2.2 Code Example

Below is some code that I wrote to familiarise myself with the Lua programming language. it outputs the lyrics to the old ”Ten Green bottles” children’s song. I got the lyrics from the [unknown, 2005, KiDiddles website].

Code Listing

```
action =" hanging on the wall"
run = 1
function greenb(x)
  for I = x,0,-1 do
    if I == 1 then
      output = (i .." green bottle" .. action)
    elseif I == 0 then
      output = ("no green bottles" .. action)
    else
      output = (i .." green bottles" .. action)
    end

    if i>0 then
      if run  = 1 then
        print(output)
        print()
      end

      run = run + 1

      print(output)
      print(output)
      print("If one green bottle should accidently fall")
      print("There’ll be...")
    else
      print(output)
    end
  end
```

```
29 end
30
31 greenb(10)
```

Breakdown of code

01 assignmeant Here is an assignmeant of a string to a variable. In this case the string is “ hanging on the wall” with the variable ‘action’.

02 assignmeant This is the assignment of an integer to the variable ‘run’. Note that there is not type definitions in the language.

03 - 29 Function declaration This is the declaration of the function ‘greenb’ which takes one argument which is referred to within the function as ‘x’.

04 - 28 for statemeant This is a for statemeant that repeats from the value of ‘x’ counting down till ‘x’ is equal to 0.

05 - 11 if statemeant This is an if statement with simple boolean comparisons featuring ‘elsif’ and ‘else’ statements

06 + 08 + 10 Concatenation of strings These lines (selected with the previous If statement) all feature another assignment of a string however this string is made up of concatenating the number stored in ‘i’ to a new string and the original ‘action’ string one useful function in the language is this concatenation using two dots ‘..’ which is fast and easy to program.

21 print() statement print is used to output a line of text (or a value) to the screen. It is very flexible and can be used for simple mathematical functions for example ‘print(3 + 4)’ would give the output ‘7’ but ‘print(“3 + 4”)’ would allow you to output the text ‘3 + 4’ if you wanted (note the addition of the quotes to signify it is a string).

31 calling a function In this instance I have included a call to the function within the code that calls the ‘greenb() function giving it the argument 10’ often you would define the functions in the file then call them from inside the interpreter or from a C program.

Lua for Config Files

In relation to the code shown in the previous section it is plain to see how it is easy to assign numbers or text to a value in a style that is verry similar to the .INI files for many applications. Lua can be built into a C (or C++) program and used to parse files that contain a .INI style layout (that is also valid Lua source) and return values to a C program.

2.3 ID3lib

this section is a library used with the information in section 2.4.2 on page 16.

[Mahoney, 2005] describes the ID3 library as:

> id3lib is an open-source, cross-platform software development library for reading, writing, and manipulating ID3v1 and ID3v2 tags. It is an on-going project whose primary goals are full compliance with the ID3v2 standard, portability across several platforms, and providing a powerful and feature-rich API with a highly stable and efficient implementation.

Calls to the ID3lib

The id3lib takes care of all the low level file I/O and also includes its own error checking. There are a number of ways in which it can be called for reading and writing. below is some sample code for accessing information in an ID3v2 frame with the library and a description of the code.

```
ID3_Tag myTag("song.mp3");

ID3_Frame *myFrame;
myFrame = new ID3_Frame;
if (NULL == myFrame)
{
  std::cout <<"\nOut of memory\n" << std::endl;
  exit(1);
}

myFrame->SetID(ID3FID_ALBUM);

ID3_Frame *pFrame;
pFrame = myTag.Find(ID3FID_ALBUM);

if (pFrame != NULL)
{
  myTag.RemoveFrame(pFrame);
}

```

```
21  if (strlen("Album Title") > 0)
22  {
23    myFrame->Field(ID3FN_TEXT) ="Album Title";
24    myTag.AttachFrame(myFrame);
25  }
26  myTag.Update();
```

1 This creaits an ID3_Tag object giving it the filename “song.mp3”.

3 - 4 This creaits a frame object (used for writing information to the tag).

5 - 9 Error checking, almost always pointles but there just incase.

11 Sets the type of field in the frame object as “ID3FID_ALBUM” (Album Title).

13 - 14 Creaits another ID3_Frame object (used for accessing the frames in the ID3_Tag object).

16 - 19 Removes any existing instances of the frame to be updated.

21 A quick string length check to make sure that data is going to be put in (there is no point having an empty frame).

23 Assigns a C string to the text field of the myFrame object.

24 Inserts the new frame into the tag.

26 The all important update() command, this actualy makes the changes (Note: no changes are made untill this is called).

2.4 Digital Audio Files & Tags

Music has been stored digitaly for a long time, however digital music on computers has only recently become common in the home with improvemeants in compression, the ease of shareing files over the internet and being able to ”burn” them to CD.

The basic concept is that a digital audio file contains the compresed data for the audio track and most of them contain optional “extra information” which is refered to as a tag. this information will often include the artist, track name, track number, album name and other similar information. the information stored and the way it is stored changes from file type to file type.

The First type of audio file I will look at are .mp3 files as these are the most prevalent on peoples computers. with information from [Nilsson, 2004] MP3 files have two methods of storing the track information. both known as “ID Tags” which are:

2.4.1 ID3v1 Tags

This is the oldest and most simple. It works by adding 128 bytes to the end of the file that contains ASCII text for the track information it works on the basis:

Song Title 30 characters

Artist 30 characters

Album Title 30 characters

Year 4 characters

Comment 30 characters

Genre 1 character

The first 3 bytes contain the letters "TAG" so that it can be identified as being there or not.

The ID3v1 format managed to not include information on the track number. Because of this it had to be adapted to hold the track number. This was done by removing the end of the comment and adding an extra field giving the following layout:

Song Title 30 characters

Artist 30 characters

Album Title 30 characters

Year 4 characters

Comment 28 characters

Blank Space 1 character (mandatory space)

Track No. 1 character

Genre 1 character

This can be read using the Java code in the appendix see section 8.3.1 on page 69 for Java code.

2.4.2 ID3v2 Tags

Information from [Mahoney, 2005] and [Nilsson, 2004]. This is more complex but circum-navigates all the shortfalls of the ID3v1 format. It works by attaching information to the start of the file and allows for variable sized fields getting past the 30 character limit. Useful when you're working with "It's The End Of The World As We Know It (And I Feel Fine)" by REM or ("It's The End Of The World As W" on ID3v1/ID3v1.1)

How the Tag works

ID3v2 Tags store information in a very different way to the ID3v1, The first difference is that the information is at the beginning of the file instead of the end. There are a large number of available fields that can be used. All the zext fields are variable length (stopping the previously mentiond problem of limited string sizes). This tag vertion also allows for images to be stored in the file.

Each aspect of the tag such as 'album name' is inserted in its own 'Frame' so a tag is just a collection of frames. Every frame contains a small heading describing its name (such as song title) its type (such as text field or image) and its size (length in bytes) as well as the relevent data. Each frame can be up to 16MB in size and the entire tag has a maximum size of 256MB, this allows for all the storage that is required for extra information (whilst you can always argue that limiting sizes paves the way for long term problems since most compressed audio files are only 3 - 5MB in size this is a more than sufficiant maximum size).

A full list of the available frames is in section 8.1 on page 64.

Due to its complexity it is far beyond the scope of this project to write my own code for accessing the information also due to the complexity of the storage of the tag information there is a library for accessing the stored information from [Mahoney, 2005]. Details for the library can be found in section 2.3 on page 13.

2.4.3 OGG Vorbis Tagging

Information from [ogg-vorbis team, 2005] Describes ogg vorbis as being "a completely open, patent-free, professional audio encoding and streaming technology with all the benefits of Open Source". It is similar to mp3 in that it stores compressed audio and additional information in a tag. It is designed so that it can be fully used instead of mp3 (advantageous as mp3 has a number of lisance laws covering it).

Similar to ID3v2 tags ogg tags are complex however there are a number of open source libraries that are provided by [ogg-vorbis team, 2005] allowing for editing of the audio content and the “extra information” which shares many of the properties of ID3v2.

2.5 Playlists

Playlists are a feature in most commonly used media players. In their most basic form playlists contain a list of files which the media playing application can work through, this is often done in the order the files appear in the playlist (so for example all the tracks could be in album order or alphabetical order) most players allow for a ‘random’ order as well. ‘XMMS’ under unix systems allows for people to search through all the files in a playlist using the ‘j’ key (jump). If a user has a collection of 100 files then it would not be hard for the player to search through all the files for matching content, however most people have many files (which don’t change regularly) and searching through all the files (including reading all the extended information) would be ineficiant and time consuming. so instad the playlist is searched. There are a number of different playlist file formats with a variety of different features. more information on playlists can be found from [Gonze, 2003].

Most playlists are propriatry formats with a few other applications supporting them. the most widely supported playlist file is m3u. Most playlist files are a simple text files containing XML (Extensible Markup Language) type layout.

2.5.1 m3u Files

m3u is the most common playlist file supported by a large number of players.

The file is (fortunatly) very simple containing ASCII text wich points to where files are stored and allows for easy editing

2.5.2 m3u File Structure

the File consists of an Identifier at the top then a list of all the files in that ‘playlist’.

Identifier

The first line of all m3u files is:

```
#EXTM3U
```

This allows for a program to check that this particular file is infact a m3u file

Description

This contains extra information for the playlist (normaly obtained from the “extra information” stored in an audio file. This information is normaly what is displayed in a media playing application when it lists all the files in the playlist(this also allows for the media playing application to not have to load information from potentialy thousends of files to display their properties.

The layout of this line in the file is:

```
#EXTINF:<length>,<name>
```

the “#EXTINF:” declaires that this is a new item in the playlist and it is followed by two attributes:

length this is the length of the file in seconds

name this is a name to display normaly in the format <artist> - <track> but can vary, it is puerly a text string so can potentialy contain anything you want

Filename

This simply contains the filename(including if necassary the relative path).

```
/home/Luser/My Music/bd-going going gone.mp3
```

Example

below is a small example of a m3u file:

```
#EXTM3U
#EXTINF:184,Bob Dylan - All Along The Watchtower
/mp3/BD-Watchtower.mp3
#EXTINF:165,Bob Dylan - Blowin’ in the Wind
/mp3/Bob Dylan - Blowin’ in the Wind.mp3
#EXTINF:183,Bob Dylan - BrownEyedGirl
/mp3/mp3/Bob Dylan - BrownEyedGirl.mp3
#EXTINF:204,stealers wheal - Stuck in the middle with you
/mp3/mp3/stealers wheal - Stuck In The Middle With You.mp3
#EXTINF:192,Bob Dylan - The Times They Are A Changin’
/mp3/mp3/Bob Dylan - The Times They Are A-Changin’.mp3
```

2.5.3 pls Files

Thease files work similarly to m3u files and were developed primarily for internet radio usage.

2.6 Searching

When it comes to searching on computers most of the time you are doing pattern matching on strings (Is this string the same as that string). the way in which this is done and what constitutes as a match (including case sensativity) varies drasticaly.

2.6.1 basic pattern matching

In the wxWidgets library there are facilities for basic pattern matching, there are a number of string functions returning boolean results for pattern matching including ‘exact matching’ and ‘contains’.

2.6.2 More advanced methods of string searching

Whilst in most circumstances basic pattern matching on strings is sufficiant and it allows you to find most things you are looking for occasionaly sircumstances call for a more complex orunusual type of comparison. In the case of this project sometimes there are alternative ways of writing band or track names. The most common being abreviations.

One thing that must be thought about is that whilst searching uses a comparativly small amount of a computers resources if the search is performed on a large number of files (1,000 or more) and the search involves many alterations on the search string to find more matches then there is a chance that the search can take too long and become ineficiant (however this is unlikely with the processing power of modern computers but must be remembered when being implemeanted).

Abreviations (full stops & Initialisation)

Often long band names (and other things) are abreviated to save typing, for example the band “Rage Against The Machine” is often shortened to “RATM” or “R.A.T.M.” so within the search algorithm it may be worth including an opertunity to initialise the given word so that when a user searches for “rage against the machine” the program also searches for “RATM” and “R.A.T.M.” also this can be expanded to “R A T M” and many other methods which can be easaly added provided the algorithm (and indeed the code for it) is flexable and makes use of good program design and where required object oriented design.

Fuzzy Matching

The main purpos of fuzzy matchin (and Fuzzy logic which it is derived from) is the idea of getting something similar. Fuzzy logic extends from “true” or “false” in that it provides a percentage answer such as “it is sunny today” this can be true or it can rain all day in which case t is false or it can rain half the day in which case it is 50% true.

Fuzzy matching commonly involves the idea of using similar words such for example if “match” is searched for “matchstick” may also be included along with “matching” (only one of the extra words will be potentialy usefull). annother common use for this is within spell checking software for word processors and text editors.

The main use of fuzzy matching within my project would be to find files with incorrectly spelt/typed names. If for example someone searched for “Dylan” and the artist on the file was “Bob Dyllan” (note the incorrect spelling) then the match would not be found. One aspect of the searching method would have to include removing letters.

2.7 Launching Applications

One area of functionality that I would like to incorporate into my project is the ability to launch third party applications including media players and file browsers. The functionality could also be used for other applications for more advanced (and customisable) functionality. One thing that makes this task difficult is that launching applications is down to the operating system and so this can be a complex area, I have researched the Unix and windows method of doing this below.

2.7.1 Unix (Fork & Exec)

The common way of launching another process under Unix involves using the ‘fork’ and ‘exec’ commands. The fork command recreates a duplicate copy of the running process in memory and exec starts a new process running in that memory space.

2.7.2 Windows ()

Windows operates differently to Unix so to launch applications under windows in C++ you have to use the ‘CreateProcess’ command.

2.7.3 WxWidgets Solution

Fortunately in part of the extra features it provides for platform independence WxWidgets provides some functions for running programmes (possibly as wrappers for the existing system calls).

2.8 CD-DB

2.8.1 Description

a CD-DB (Compact Disk Database) is a database of Cd's that can be easily referenced to get information about a CD and its tracks that is most commonly used to add information to the tags of digital audio files when a CD is being "ripped". however these databases can be accessed via web forms or a custom protocol and searched

2.8.2 Examples

Here are some CD-DB websites (note both allow free searching of their databases):
Gracenote - `www.cddb.com` (CD-DB with a commercial approach)

freedb.org `http://www.freedb.org/` (CD-DB with a "Free Software" approach)

My focus is on freedb as I am a fan of the "Free Software" ideology. This can be accessed by two methods, There is a specific protocol (CDDcreatedBP) that can be used to access the database or there is a web interface.

2.8.3 CDDBP

This is the protocol created to access the database. Its design is similar to the HTML protocol in that it uses number codes in the return fields to distinguish the type of response. Below I have listed the console of a telnet session where I tested out some of the features of the protocol and underneath that is a short description of the calls.

Telnet

```
user@computer: $ telnet freedb.freedb.org 8880
Trying 64.71.163.204...
Connected to freedb.org.
Escape character is '^]'.
201 zaphod CDDBP server v1.5.1PL0 ready at Fri Mar 18 14:13:19 2005
cddb hello davidhalliday abc.fubar.com mp3org 0.1
200 Hello and welcome davidhalliday@abc.fubar.com running mp3org 0.1.
cddb lscat
210 OK, category list follows (until terminating '.')
data
```

```
11  folk
12  jazz
13  misc
14  rock
15  country
16  blues
17  newage
18  reggae
19  classical
20  soundtrack
21  .
22  motd
23  210 Last modified: 04/02/2001 12:00:00 MOTD follows (until
terminating ‘.’)
24  Welcome to freedb.org.
25  You can find the freedb website at http://www.freedb.org
26  If you have questions or suggestions write to info@freedb.org
27  .
28  sites
29  210 OK, site information follows (until terminating ‘.’)
30  freedb.freedb.org 8880 N000.00 W000.00 Random freedb server
31  at.freedb.org 8880 N048.13 E016.22 Vienna, Austria
32  au.freedb.org 8880 S033.52 E151.13 Sydney, Australia
33  ca.freedb.org 8880 N049.48 W097.08 Winnipeg, MB Canada
34  ca2.freedb.org 8880 N045.28 W073.45 Montreal, QC Canada
35  de.freedb.org 8880 N052.53 E013.31 Berlin, Germany
36  es.freedb.org 8880 N040.30 W003.48 Madrid, Spain
37  fi.freedb.org 8880 N061.30 E023.42 Tampere, Finland
38  ru.freedb.org 8880 N059.55 E030.15 Saint-Petersburg, Russia
39  uk.freedb.org 8880 N051.49 W000.01 London, UK
40  us.freedb.org 8880 N037.21 W121.55 San Jose, CA USA
41  .
42  quit
43  230 zaphod Closing connection.  Goodbye.
44  Connection closed by foreign host.
```

Description

1 - 3 terminal and telnet commands/messages.

4 - 5 welcome and configuration messages from the server.

6 client side of the “Handshake” consisting of `cddb hello <username> <hostname> <clie`

7 server reply of the “Handshake”.

8 client command to list categories(genres) on the server.

9 - 21 list of categories(genres). on the server.

22 motd client call (Message of the day).

23 - 27 the motd (message of the day).

28 sites command. This command gets the server to return a list of available official mirrors.

29 - 41 returned list of official mirrors.

42 quit command to disconect from server.

43 server confirmation of sisconect.

44 telnet message to say that connection has finished.

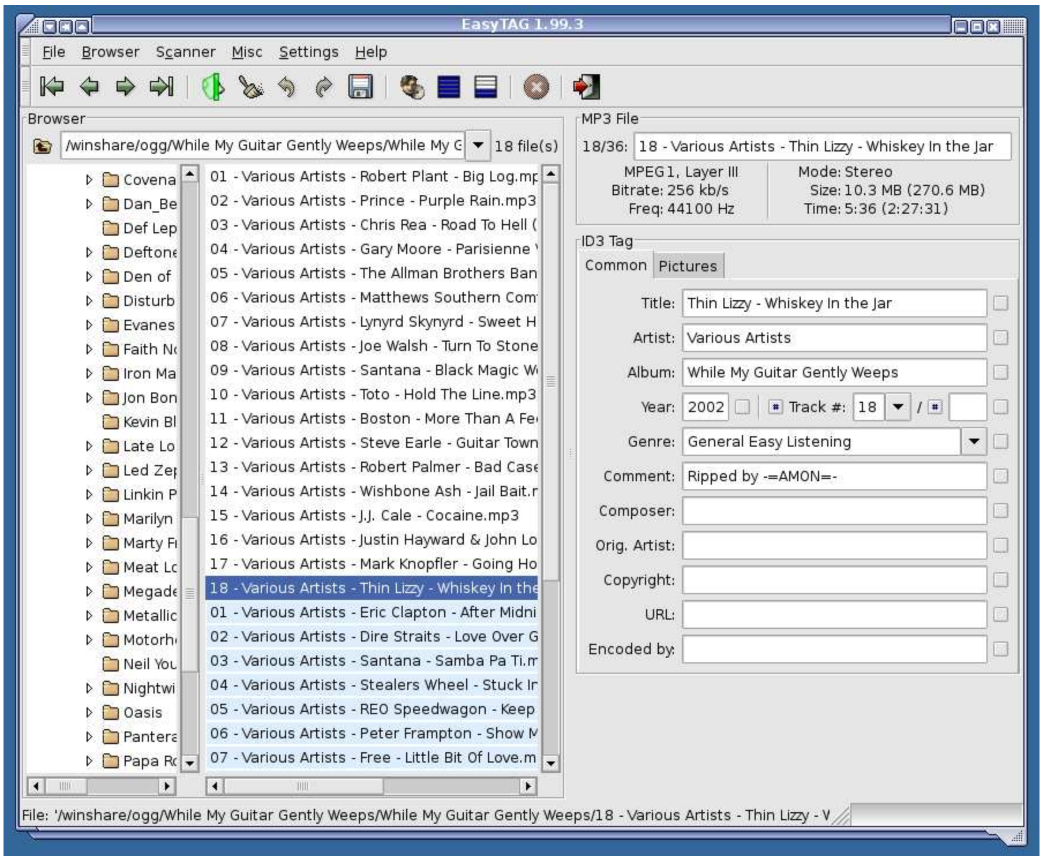
2.8.4 Web Interface

This includes options to search for band, album and track names or a multitude of other things to find the information for the CD you are looking for. This is accessed through a cgi script that is used by a number of applications to retrieve lists of matches. the difficulty of this approach is that the client program has to have a basic HTTP client module and be able to parse the returned HTML which is beyond the scope of this project.

2.9 Existing Applications

As part of my research I have looked at a number of applications that have functionality in the areas I am interested in implementing. I have also looked into applications that work in similar ways to how i want my application to function.

2.9.1 EasyTAG



/winshare/ogg/While My Guitar Gently Weeps/While My G 18 file(s)

Covena
Dan_Be
Def Lep
Deftone
Den of
Disturb
Evanes
Faith No
Iron Ma
Jon Bon
Kevin Bl
Late Lo
Led Zep
Linkin P
Marilyn
Marty Fi
Meat Lo
Megade
Metallic
Motorh
Neil You
Nightwi
Oasis
Pantera
Papa Ro

01 - Various Artists - Robert Plant - Big Log.mp
02 - Various Artists - Prince - Purple Rain.mp3
03 - Various Artists - Chris Rea - Road To Hell (
04 - Various Artists - Gary Moore - Parisienne '
05 - Various Artists - The Allman Brothers Ban
06 - Various Artists - Matthews Southern Com
07 - Various Artists - Lynyrd Skynyrd - Sweet H
08 - Various Artists - Joe Walsh - Turn To Stone
09 - Various Artists - Santana - Black Magic W
10 - Various Artists - Toto - Hold The Line.mp3
11 - Various Artists - Boston - More Than A Fe
12 - Various Artists - Steve Earle - Guitar Town
13 - Various Artists - Robert Palmer - Bad Case
14 - Various Artists - Wishbone Ash - Jail Bait.r
15 - Various Artists - J.J. Cale - Cocaine.mp3
16 - Various Artists - Justin Hayward & John Lo
17 - Various Artists - Mark Knopfler - Going Ho
18 - Various Artists - Thin Lizzy - Whiskey In the
01 - Various Artists - Eric Clapton - After Midni
02 - Various Artists - Dire Straits - Love Over G
03 - Various Artists - Santana - Samba Pa Ti.m
04 - Various Artists - Stealers Wheel - Stuck In
05 - Various Artists - REO Speedwagon - Keep
06 - Various Artists - Peter Frampton - Show M
07 - Various Artists - Free - Little Bit Of Love.m

18/36: 18 - Various Artists - Thin Lizzy - Whiskey In the Jar

MPEG1, Layer III Mode: Stereo
Bitrate: 256 kb/s Size: 10.3 MB (270.6 MB)
Freq: 44100 Hz Time: 5:36 (2:27:31)

Title: Thin Lizzy - Whiskey In the Jar
Artist: Various Artists
Album: While My Guitar Gently Weeps
Year: 2002 Track #: 18 /
Genre: General Easy Listening
Comment: Ripped by -=AMON=-
Composer:
Orig. Artist:
Copyright:
URL:
Encoded by:

File: '/winshare/ogg/While My Guitar Gently Weeps/While My Guitar Gently Weeps/18 - Various Artists - Thin Lizzy - V

This is a common program on a number of Linux distributions, It has many features. I personaly found the user interface a little confusing to at first. When you first load the program it starts searching in a users home directory for supported audio files. The programme is written in C and uses GTK+ 2.4 for the GUI.

Supported File Formats

mp3

mp2

Ogg Vorbis

FLAC

Muse Pack

Monkeys audio files

Features

Here is a list of some of the significant programme features, most of the information is from [easyTAG team, 2005]

CD-DB support This is done using Freedb.org servers (manual and automatic search) To get Extended information for files from the internet.

View, edit, write tags Some basic functions for editing the extended information on the supported file types.

Auto tagging This is acheaved by parseing the file name and directory to automatically complete the fields (using masks).

rename files The ability to rename files from the tag (using masks) or by loading a text file.

Read file header information (bit rate, time, ...) and display them (most of this is covered by the various libraries and relevent appendecies).

Launch third party applications Opens up a user specified media player with a selected track (or tracks) for playing.

A list to select files This is a common Feature of similar applications, all the files are displayed in a list for easy selection.

play list generator This is a nice feature for adding files to supported playlist files, for information on playlists see section 2.5 on page 17.

file searching This is simple pattern matching on files including File names and extended information.

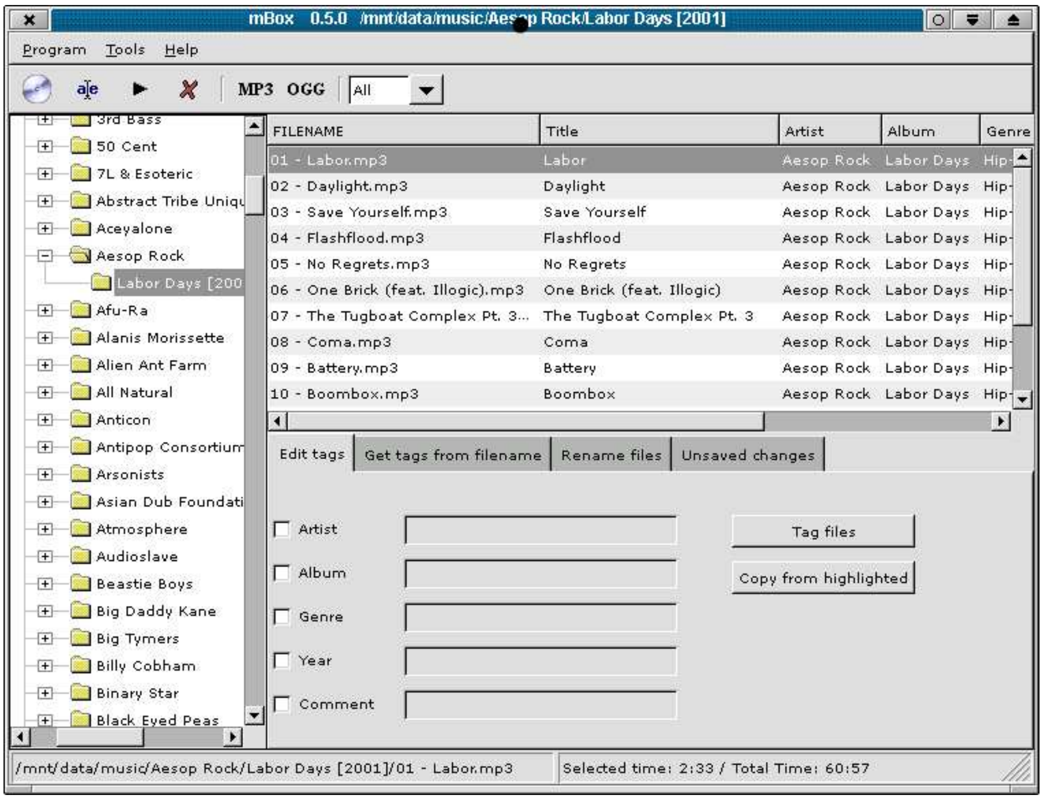
translation languages The programme is available in the folloing languages French, German, Russian, Dutch, Hungarian, Swedish, Italian, Japanese, Ukrainian, Czech, Spanish, Polish Romanian and Danish

Limitations

Not all tags supported Can edit only the following tag fields : Title, Artist, Album, Year, Track Number, Genre and Comment. (Most ID3v2 tag fields aren't supported see section 2.4.2 on page 16 for details on ID3v2 tags).

Confusing GUI There is a lot going on on the screen when you first load the application which can confuse some users.

2.9.2 mBox



FILENAME	Title	Artist	Album	Genre
01 - Labor.mp3	Labor	Aesop Rock	Labor Days	Hip-
02 - Daylight.mp3	Daylight	Aesop Rock	Labor Days	Hip-
03 - Save Yourself.mp3	Save Yourself	Aesop Rock	Labor Days	Hip-
04 - Flashflood.mp3	Flashflood	Aesop Rock	Labor Days	Hip-
05 - No Regrets.mp3	No Regrets	Aesop Rock	Labor Days	Hip-
06 - One Brick (feat. Illogic).mp3	One Brick (feat. Illogic)	Aesop Rock	Labor Days	Hip-
07 - The Tugboat Complex Pt. 3...	The Tugboat Complex Pt. 3	Aesop Rock	Labor Days	Hip-
08 - Coma.mp3	Coma	Aesop Rock	Labor Days	Hip-
09 - Battery.mp3	Battery	Aesop Rock	Labor Days	Hip-
10 - Boombox.mp3	Boombox	Aesop Rock	Labor Days	Hip-

/mnt/data/music/Aesop Rock/Labor Days [2001]/01 - Labor.mp3   Selected time: 2:33 / Total Time: 60:57

mBox is a utility primarily for converting audio files from mp3/wma/ogg to mp3/ogg but has good features for tag editing. Most of the information here is from [Hnilica, 2005].

Supported File Formats

- mp3
- wma
- Ogg Vorbis

Features

re encode files The ability to re encode mp3/ogg/wma to mp3/ogg. This is the primary function of the application. Note that the programme does not support writing to WMA files.

reading/writing tags Some basic functions for editing the extended information on the supported file types.

FreeDB (CD-DB) supports FreeDB, treats encoded files as if it was CD, so you can simply select your files and - if found in database - write tags to them.

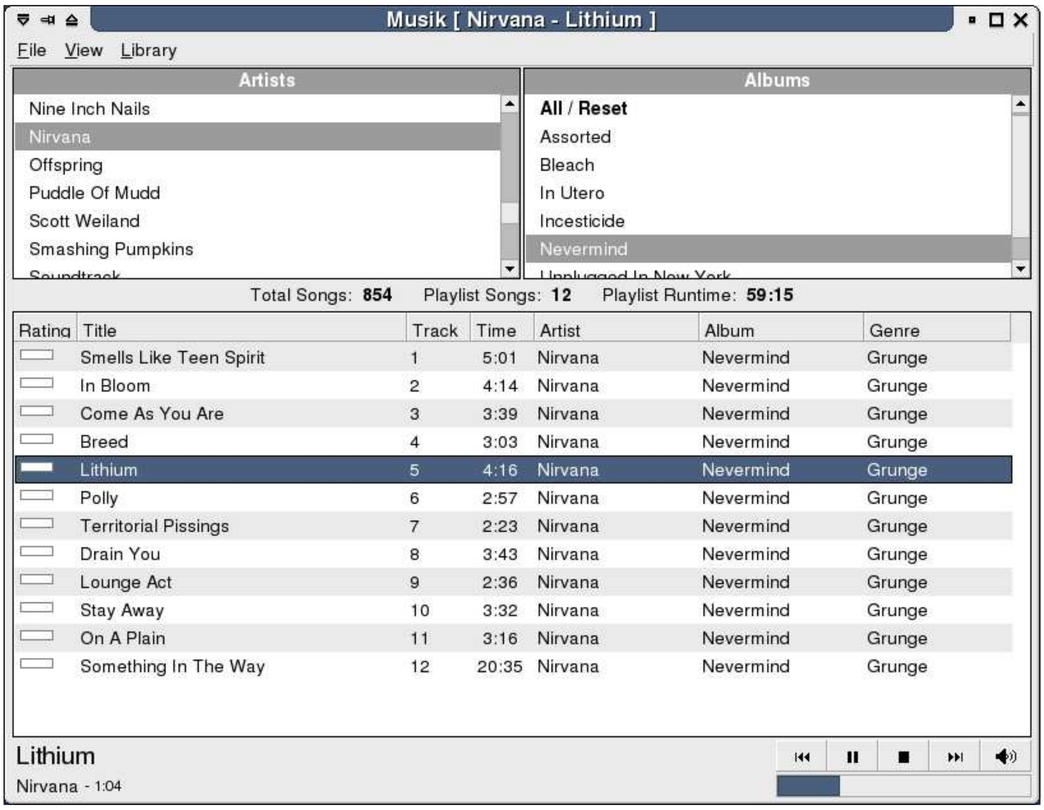
Multiple (batch) tag editing This allows a user to simultaneously edit tags on multiple files.

Limitations

Lack of files only 3 file types supported (although I assume more could be added if required).

Lack writing to wma tags This again is a small problem as the programme is primaraly concerned with converting from wma files to other better supported file formats.

2.9.3 WXmusik



Artists: Nine Inch Nails, Nirvana, Offspring, Puddle Of Mudd, Scott Weiland, Smashing Pumpkins, Soundtrack

Albums: All / Reset, Assorted, Bleach, In Utero, Incesticide, Nevermind, Unplugged In New York

Total Songs: **854** Playlist Songs: **12** Playlist Runtime: **59:15**

Rating	Title	Track	Time	Artist	Album	Genre
▭	Smells Like Teen Spirit	1	5:01	Nirvana	Nevermind	Grunge
▭	In Bloom	2	4:14	Nirvana	Nevermind	Grunge
▭	Come As You Are	3	3:39	Nirvana	Nevermind	Grunge
▭	Breed	4	3:03	Nirvana	Nevermind	Grunge
▭	Lithium	5	4:16	Nirvana	Nevermind	Grunge
▭	Polly	6	2:57	Nirvana	Nevermind	Grunge
▭	Territorial Pissings	7	2:23	Nirvana	Nevermind	Grunge
▭	Drain You	8	3:43	Nirvana	Nevermind	Grunge
▭	Lounge Act	9	2:36	Nirvana	Nevermind	Grunge
▭	Stay Away	10	3:32	Nirvana	Nevermind	Grunge
▭	On A Plain	11	3:16	Nirvana	Nevermind	Grunge
▭	Something In The Way	12	20:35	Nirvana	Nevermind	Grunge

Lithium
Nirvana - 1:04

This application is more of a media player than for editing tags, however it has got a fully integrated tag editing facility. This application is made using the WxWidgets library and is platform independent (with a few feature differences between versions for different platforms such as support for wma (not available under Linux). Most of the information here is from [Langen, 2005].

Supported File Formats

- mp3
- mp2
- Ogg Vorbis
- wav
- FLAC
- Muse Pack
- Monkeys audio files
- aiff
- APE
- MPC
- wma (windows only)

Features

Net streaming support for icecast and shoutcast streams (internet radio broadcasts).

batch and auto-tagging This allows a user to simultaneously edit tags on multiple files.

Embedded SQL database This makes searching and organizing 1000's of files quick and easy.

keyboard play control keys Supports the "multimedia" control keys that are on some modern keyboards .

play list creation Using drag and drop files can be added to playlists. See section 2.5 on page 17 for information on playlists.

Localized in 8 languages The programme is available in the folloing languages German, Czech, Italian, Spanish, Portuguese, Norwegian, french and dutch.

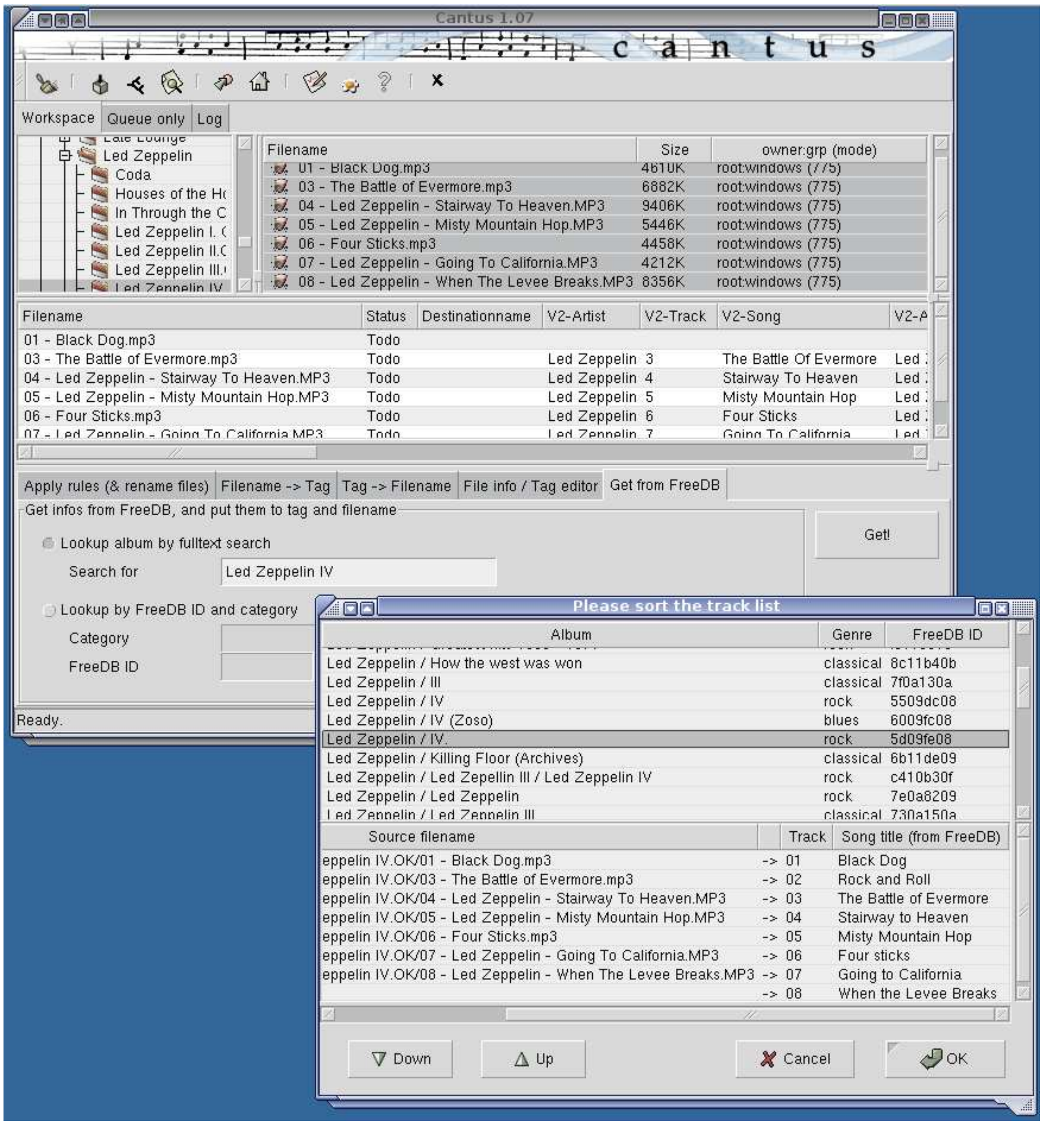
Fuzzy search feature This allows for string matching in the event of misspellings in file names and extended information. See section 2.6 on page 19 for information on playlists.

album art display Displays the cover of currently playing album (if available).

Limitations

I have not tested this application and have not found any limitations with it in terms of support or features.

2.9.4 Cantus



Filename	Size	owner:grp (mode)
01 - Black Dog.mp3	4610K	root:windows (775)
03 - The Battle of Evermore.mp3	6882K	root:windows (775)
04 - Led Zeppelin - Stairway To Heaven.MP3	9406K	root:windows (775)
05 - Led Zeppelin - Misty Mountain Hop.MP3	5446K	root:windows (775)
06 - Four Sticks.mp3	4458K	root:windows (775)
07 - Led Zeppelin - Going To California.MP3	4212K	root:windows (775)
08 - Led Zeppelin - When The Levee Breaks.MP3	8356K	root:windows (775)

Filename	Status	Destinationname	V2-Artist	V2-Track	V2-Song	V2-A
01 - Black Dog.mp3	Todo					
03 - The Battle of Evermore.mp3	Todo		Led Zeppelin	3	The Battle Of Evermore	Led
04 - Led Zeppelin - Stairway To Heaven.MP3	Todo		Led Zeppelin	4	Stairway To Heaven	Led
05 - Led Zeppelin - Misty Mountain Hop.MP3	Todo		Led Zeppelin	5	Misty Mountain Hop	Led
06 - Four Sticks.mp3	Todo		Led Zeppelin	6	Four Sticks	Led
07 - Led Zeppelin - Going To California.MP3	Todo		Led Zeppelin	7	Going To California	Led

Get infos from FreeDB, and put them to tag and filename

Lookup album by fulltext search

Search for: Led Zeppelin IV

Lookup by FreeDB ID and category

Category

FreeDB ID

Ready.

Album	Genre	FreeDB ID
Led Zeppelin / How the west was won	classical	8c11b40b
Led Zeppelin / III	classical	7f0a130a
Led Zeppelin / IV	rock	5509dc08
Led Zeppelin / IV (Zoso)	blues	6009fc08
Led Zeppelin / IV.	rock	5d09fe08
Led Zeppelin / Killing Floor (Archives)	classical	6b11de09
Led Zeppelin / Led Zeppelin III / Led Zeppelin IV	rock	c410b30f
Led Zeppelin / Led Zeppelin	rock	7e0a8209
Led Zeppelin / Led Zeppelin III	classical	730a150a

Source filename		Track	Song title (from FreeDB)
eppelin IV.OK/01 - Black Dog.mp3	->	01	Black Dog
eppelin IV.OK/03 - The Battle of Evermore.mp3	->	02	Rock and Roll
eppelin IV.OK/04 - Led Zeppelin - Stairway To Heaven.MP3	->	03	The Battle of Evermore
eppelin IV.OK/05 - Led Zeppelin - Misty Mountain Hop.MP3	->	04	Stairway to Heaven
eppelin IV.OK/06 - Four Sticks.mp3	->	05	Misty Mountain Hop
eppelin IV.OK/07 - Led Zeppelin - Going To California.MP3	->	06	Four sticks
eppelin IV.OK/08 - Led Zeppelin - When The Levee Breaks.MP3	->	07	Going to California
	->	08	When the Levee Breaks

This is the description of Cantus provided by [Abels, 2005].

> Cantus is an easy to use tool for tagging and renaming MP3 and OGG/Vorbis files. It has many features including mass tagging and renaming of MP3s, the ability to generate a tag out of the file name, filter definitions for renaming, recursive actions, CD-DB (Freedb) look up (no CD needed), copy between ID3V1 and ID3V2 tags, and a lot more.

Although not originally platform independent there is a project that ports the latest version to windows.

This program appears to have a small number of features, however it does the small list of features better than any other programmes I have seen, it is easy to use and has all the features you regularly need.

Supported File Formats

- mp3
- Ogg Vorbis

Features

edit/create tags Some basic functions for editing the extended information on the supported file types.

CD-DB support supports FreeDB, treats encoded files as if it was CD, so you can simply select your files and - if found in database - write tags to them.

multiple renaming This allows a user to simultaneously edit tags on multiple files.

file renaming This allows files to be renamed from the extended information.

Limitations

Not many files supported Only two file formats are supported, however this is natively a Unix application and thease are the two most common file types.

Not fully supporting ID3v2 Only supports the comparative fields with ID3v1 & ID3v1.1 See section 2.4.2 on page 16 for information on id3v2 tags.

Chapter 3

Project Specification

After looking into existing applications I have got an idea of what I want my application to do and how I want to do it. Here I'll look at what file types I want to support, some of the basic features I want to fit in my program then move on to thinking about the design and development model I want to use.

3.1 Supported File Formats

Primarily I am only designing for mp3 files (using id3 tags see sections 2.4.1 on page 15 and 2.4.2 on page 16) as they are the most common, however I will design my program to be modular (with the use of Object Oriented programming) so that further file formats (and new tag formats) can be used such as Ogg Vorbis files (see section 2.4.3 on page 16).

3.2 Features

Here are some basic features I want to implement into my application.

Edit/create tags Some basic functions for editing the extended information on the supported file types.

Multiple renaming To allow the user to perform multiple changes to the extended information in files simultaneously (on multiple files).

File renaming To allow files to be renamed from the tag information.

File copying The ability to copy files from one place to annother.

Play files The ability to launch a predefined media player (such as xmms or winamp) and play selected files.

Sort files The ability to organise files by the extended information such as by band name and album name.

Search feature The ability to search through a list of files for the desired audio file to be edited.

3.3 Design aims

3.3.1 Make Platform Independent

I want my program to be able to run on all the common operating systems (windows, Unix, Linux, Mac). This means I have to find a language that will allow me to develop it in such a manner. Bearing in mind that many functions of the application (GUI & File I/O) are very platform dependent.

3.3.2 easy & efficient GUI

My aim to have a GUI application is to make the functions easier to access from the user than say a command line application. The GUI has to be easy and efficient (or there is little point to it) with the important and commonly used functions easily accessible through menus and buttons.

3.3.3 Easaly updatable

One feature i am keen to implemeant into my project is to make it easy to update and edit. This will make the programme easier to add new functionality to and in the long term speed up developmeant. Annother advantage is that once the application is a reasonable size it will make debugging easier (problems can be easily locked down to a small section of code). I plan to do this by making good use of object oriented programming techniques.

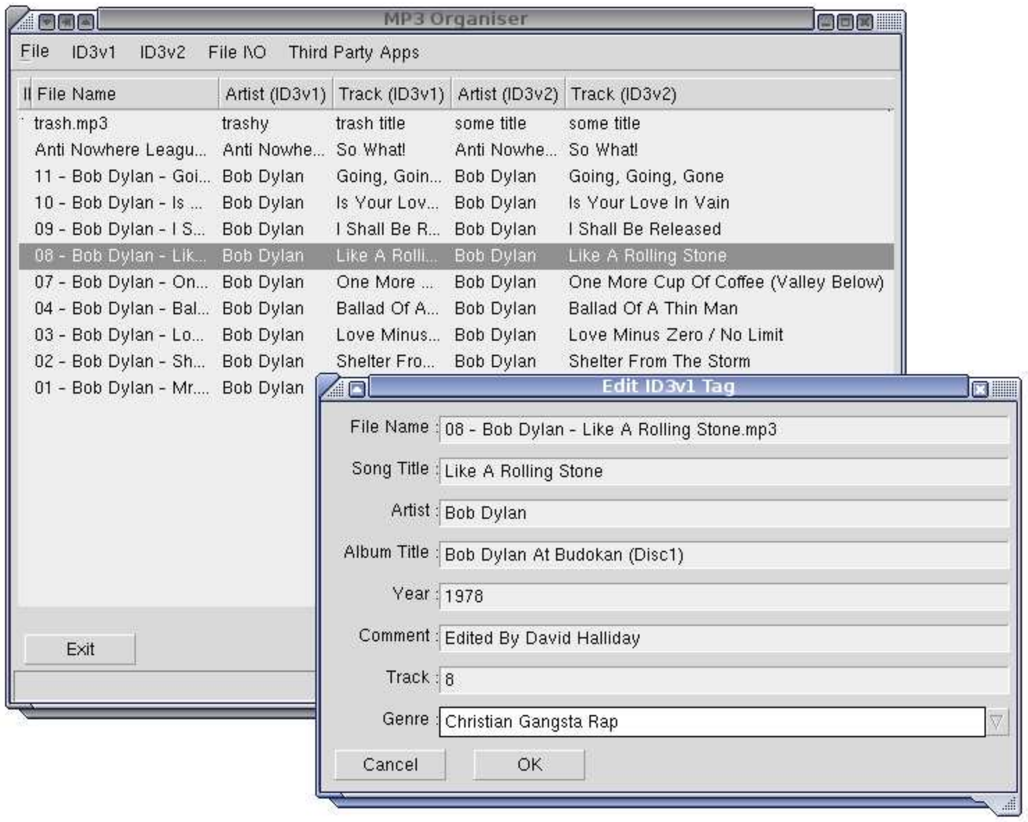
Chapter 4

Architecture & Class design

This is a full description of my program and how it works starting off with a screen dump of the application running and a short guide on how to do some of the simplest/most common functions in the application moving on through the files used in the application and including information on how some of the classes are called and used.

4.1 Screen Dump of User Interface

Below is a screen dump of the GUI application running on a computer running Linux with X11. It shows the application with a list of mp3 files one of which is selected, and the tag is being edited in a separate dialog.



File Name	Artist (ID3v1)	Track (ID3v1)	Artist (ID3v2)	Track (ID3v2)
trash.mp3	trashy	trash title	some title	some title
Anti Nowhere Leagu...	Anti Nowhe...	So What!	Anti Nowhe...	So What!
11 - Bob Dylan - Goi...	Bob Dylan	Going, Goin...	Bob Dylan	Going, Going, Gone
10 - Bob Dylan - Is ...	Bob Dylan	Is Your Lov...	Bob Dylan	Is Your Love In Vain
09 - Bob Dylan - I S...	Bob Dylan	I Shall Be R...	Bob Dylan	I Shall Be Released
08 - Bob Dylan - Lik...	Bob Dylan	Like A Rolli...	Bob Dylan	Like A Rolling Stone
07 - Bob Dylan - On...	Bob Dylan	One More ...	Bob Dylan	One More Cup Of Coffee (Valley Below)
04 - Bob Dylan - Bal...	Bob Dylan	Ballad Of A...	Bob Dylan	Ballad Of A Thin Man
03 - Bob Dylan - Lo...	Bob Dylan	Love Minus...	Bob Dylan	Love Minus Zero / No Limit
02 - Bob Dylan - Sh...	Bob Dylan	Shelter Fro...	Bob Dylan	Shelter From The Storm
01 - Bob Dylan - Mr....	Bob Dylan			

File Name :	08 - Bob Dylan - Like A Rolling Stone.mp3
Song Title :	Like A Rolling Stone
Artist :	Bob Dylan
Album Title :	Bob Dylan At Budokan (Disc1)
Year :	1978
Comment :	Edited By David Halliday
Track :	8
Genre :	Christian Gangsta Rap

4.2 Summary of Features

To show some of the functionality within the application I have included here a brief guide on how to perform some of the basic functions within my application, more in depth information similar to this can be found in the user guide in section 8.2 on page 66.

Load application

The application is launched by running the binary file ("mp3org.exe" under windows or "mp3org" under Linux). when the application loads it will read from the current working directory the configuration file(If not found default options are used).

List Files

Before you can do anything with any files you have to list all the files in the directory you wish to look at. This includes reading all the existing extended information on those files. this is done by going to the menu and selecting:

```
File>Dir
```

Then using the Directory selection box you select the directory that you want to look at the files in. and click OK. then the files will be listed in the list box on the main window of the application and some of their Tag data will be shown as well.

Edit ID3v1 Tag

Once you have got a list of file(s) in a directory you can highlight one (or more) and select from the drop down menu:

```
ID3v1>Edit Tag
```

This can also be done by right clicking on the item(s) in the list and selecting "Edit ID3v1 Tag"

Then the ID3v1 Edit dialog will be shown with the first selected file's ID3v1 tag information. Once you have made your changes to this file you can click "OK" to save them or "Cancel" to not save any changes. after clicking "OK" or "Cancel" the next selected files' tag is displayed for editing till the last tag has been edited.

Edit ID3v2 Tag

Once you have got a list of file(s) in a directory you can highlight one (or more) and select from the drop down menu:

```
ID3v2>Edit Tag
```

This can also be done by right clicking on the item(s) in the list and selecting "Edit ID3v2 Tag"

Then the ID3v1 Edit dialog will be shown with the first selected file's ID3v1 tag information. Once you have made your changes to this file you can click "OK" to save them or "Cancel" to not save any changes. after clicking "OK" or "Cancel" the next selected files' tag is displayed for editing till the last tag has been edited.

Batch Edit ID3v1 Tag

To edit multiple tags simultaneously (eg. set all files to the same artist) select the desired files to edit in the same way as before and select from the drop down menu:

```
ID3v1>Batch Edit Tag
```

This opens up a dialog similar to before except that this time no information is displayed and any fields you add text to will overwrite all the same fields in the file (eg. if you change the contents of the artist box all the files will have their artist updated and whatever was there before will be replaced). any fields left blank will not be altered

Again you can click "OK" to perform the changes or "Cancel" to not make any changes. there is only one dialog here that changes all selected files.

Copy out files

Once you have finished editing tags with the application, you can then select all the files you want to sort and the program will copy them to a location of your choice with the directory and file name structure:

```
/<BAND NAME>/<ALBUM NAME>/<TRACK NUMBER> - <TRACK NAME>.mp3
```

To do this select the files as normal and select from the drop down menu:

```
FileI\O>Sort Files
```

Play files

To play selected files in a media player (selected by you in the configuration file) select the files as normal and select from the drop down menu:

```
Third Party Apps>Play Selected Files
```

Add To Play List

There is a facility to add files to a m3u play list (the play list has to be referenced in the configuration file). to do this select the files as normal and select from the menu that appears when you right click on the list:

```
Add to play list>NAME OF PLAYLIST
```

Note that the name of the play lists will be shown and you click on the play list you want to add the file to.

Rename with/without preceding track name

This feature was added due to user feedback, Some users like to organise their files in a directory by track number so that a media player will automatically list the files in a play list in album order instead of alphabetical order (other users hate having the track number on the file name) so for this I have created functionality to add or remove the track number to files. this is done by selecting the files as normal and selecting from the menu (Respectively):

`File I\O>Rename With Track No.` ← To add the track number

`File I\O>Rename With Track No. Removed` ← To remove the track number

This similar to sorting files allows you to copy files. the files are copied rather than being edited directly allows the user to check the new files to save original files being edited wrongly or corrupted.

4.3 File Layout

This is a list of all the files in the application (not including the makefile which is shown in section 8.3.2 on page 70). The code in these files starts In section 8.3.3 on page 70. The files below are all my own code with exceptions in genre.cpp and id3v2.cpp where I have modified code from a GNU project.

mp3org.cpp This is the central c++ file that contains includes for the other files, The 'OnInit' and 'IMPLEMENT_APP(mp3org)' see section 2.1.3 on page 8 for information on these important aspects of using the WxWidgets library.

MainFrame.h Headers for the MainFrame class

MainFrame.cpp functions for the GUI controls and calling lower level functions.

m3uFile.cpp the m3uFile class for manipulation of m3u play list files (each object points to a separate file).

AudioFile.h Headers for the AudioFile class

AudioFile.cpp This class contains information on an individual audio file and subclasses of this are id3v1 and id3v2 which interface with the different information stored within the files.

id3v1.h Headers for the id3v1 class

id3v1.cpp This class contains the lower level functions and library calls for extracting and writing the id3v1 tag to and from mp3 files.

id3v2.h Headers for the id3v2 class

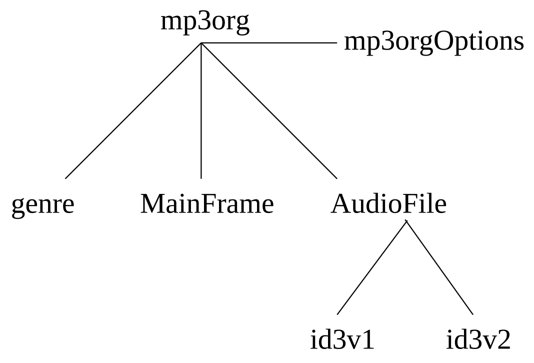
id3v2.cpp This class contains the lower level functions and library calls for extracting and writing the id3v2 tags to and from mp3 files.

genre.cpp this is a collection of functions to match text strings of genres to the numbers stored in the id3v1 tag.

mp3orgoptions.cpp this is a class that interacts with the options file. often called from other classes to get various configurable settings including applications to launch and initial size of the application window.

4.4 Class Hierarchy

Below is a diagram showing how different classes have a hierarchy in calling each other.



4.4.1 Role & Behaviour of classes

The entire application is a Graphical User Interface application with event driven calls to subroutines for file Input/Output. The user interface is in the ‘mainframe’ class, It is called from the equivalent of the “main” function in the code. All the other classes are separate from the GUI and return information to the GUI. The actual File I/O for editing files takes place in the ID3v1 & ID3v2 classes.

mp3orgOptions

This class is responsible for reading the options file (mo3org.conf) and making the options in that file available to the rest of the application.

MainFrame

This class is responsible for all the GUI functions and processing of the events when a user performs an action such as clicking on a button or selecting an item from the list of audio files. Many of the functions within this class revolve around getting information from the user to perform functions on files selected from the list, to see some examples of this see see section 4.2 on page 35. The functions work by parsing through all the selected items in the list then performing actions on them which primarily involve calls to other classes.

genre

This is a small class with functions to get the text string associated with a genre number which is stored in an id3v1 tag, it also converts strings given to it to a number to be stored as a genre.

AudioFile

This class is primarily a wrapper for the id3v1 and id3v2 classes so that it is easy to treat any type of audio file (with supported extended information) as being the same in an index which is a vector of AudioFile objects.

id3v1

This class performs all the File i/o and error checking of data for id3v1 tags within mp3 files. unlike the id3v2 class none of the functions are performed by an external library.

id3v2

This class performs all the File i/o and error checking of data for id3v2 tags within mp3 files. Since i am using a library (id3lib) to access the id3v2 tags this class acts more like a wrapper for the library than performing tasks itself.

4.5 Class & Function calls

This documentation covers How the various functions are called from one class to another. I have separated the calls into their respective classes.

4.5.1 mp3orgOptions

This class is involved with parsing an options file (mp3org.conf) and extracting settings from there and making settings available to other parts of the program.

mp3orgOptions()

mp3orgOptions() This is the constructor, it takes no arguments and creates an options object and fills it with options from the options file.

GetBrowser()

wxString GetBrowser() This function takes no arguments and returns a wxString containing the browser specified in the options file.

GetPlayer()

wxString GetPlayer() This function takes no arguments and returns a wxString containing the media player specified in the options file.

GetProgrammeWidth()

long GetProgrammeWidth() This function takes no arguments and returns a long (integer) containing the application width specified in the options file.

GetProgrammeHeight()

long GetProgrammeHeight() This function takes no arguments and returns a long (integer) containing the application height specified in the options file.

PlaylistVector

PlaylistVector This is the only case in the application where a data structure in a class is defined as public. This is a vector that contains the list of play list files extracted from the options file that mp3s can be added to.

4.5.2 MainFrame

although mainframe has many functions within it for the the GUI and calls to other functions it only has one public function which is the constructor.

MainFrame()

MainFrame(const wxString title, **const wxPoint** pos, **const wxSize** size) This is the constructor, it takes three arguments to create the GUI and display it on the screen.
Parameters

title This is the title at the top of the application window.

pos This takes a specific data type used in WxWidgets for positioning and defines where on the screen the top left corner of the application loads.

size This is similar to the position in that it takes two numbers in a specific WxWidgets data type and uses them to define the width and height of the application window.

4.5.3 genre

The Genre class is used for editing the genres in ID3v1 tags due to there only being a number stored in the tag and that number has to be matched to a string.

genre()

genre() This is the constructor, it takes no arguments.

GetGenreFromNum()

wxString GetGenreFromNum(int GenreNo**)** This function returns the genre string that matches the given number.
Parameters

GenreNo This is the genre number taken from the ID3v1 tag.

GetNumFromGenre()

int GetNumFromGenre(wxString GenreName**)** This function returns the genre string that matches the given number.
Parameters

GenreName This is a string containing a genre that needs to be matched to a number to be saved in a ID3v1 tag.

GetGenreCount()

int GetGenreCount() This function returns the number of possible genres.

4.5.4 AudioFile

This class is primarily a wrapper for the ID3v1 and ID3v2 subclasses and allows for other audio file and tagging formats to be supported.
Note: ID3v1 & ID3v2 are subclasses of AudioFile

AudioFile()

AudioFile(wxString GivenFileName**, wxString** GivenDir**)** This is the constructor, it takes two arguments.
Parameters

GivenFileName This is the file name of the audio file.

GivenDir This is the full directory for the file.

HasID3v1Tag()

bool HasID3v1Tag() This checks to see if the file has an ID3v1 Tag and returns true if there is an ID3v1 and false if not.

MakeId3v1()

bool MakeId3v1() This function creates an ID3v1 tag if one does not exist already and returns true if successful.

GetFileName()

wxString GetFileName() This gets the wxString containing the file name.

GetDir()

wxString GetDir() This gets the wxString containing the directory name.

GetPathAndFileName()

wxString GetPathAndFileName() This gets the wxString containing the directory and file name.

ThisID3v1

This is a pointer to the ID3v1 object.

ThisID3v2

This is a pointer to the ID3v2 object.

4.5.5 ID3v1

ID3v1 deals with all the File I/O for the ID3v1 tags.
Note: ID3v1 & ID3v2 are subclasses of AudioFile

ID3v1()

ID3v1(wxString GivenFileName**, wxString** GivenDir**, int state)** This is the constructor, it takes three arguments.
Parameters

GivenFileName This is the file name of the audio file.

GivenDir This is the full directory for the file.

state This is a numeric value - 0 normally or 1 for the creation of a new ID3v1 tag note this is done this way to leave room for future modes.

GetTag()

wxString GetTag() This gets the wxString containing the full tag.

GetSongTit()

wxString GetSongTit() This gets the wxString containing the Song Title.

GetArtist()

wxString GetArtist() This gets the wxString containing the Artist Name.

GetAlbumTit()

wxString GetAlbumTit() This gets the wxString containing the Album Title.

GetYear()

wxString GetYear() This gets the wxString containing the Year.

GetComment()

wxString GetComment() This gets the wxString containing the Comment.

GetTrackNo()

int GetTrackNo() This gets the int containing the Track Number.

GetGenre()

int GetGenre() This gets the int containing the genre.

WriteTag()

bool WriteTag(wxString NewTag**)** This writes the new given Tag and returns true if successful.
Parameters

NewTag This is the full 128b tag

WriteSongTit()

bool WriteSongTit(wxString NewSongTit**)** This writes the new given Song Title and returns true if successful.
Parameters

NewSongTit This is the new song title to be written to the tag.

WriteArtist()

bool WriteArtist(wxString NewArtist**)** This writes the new given artist and returns true if successful.
Parameters

NewArtist This is the new artist to be written to the tag.

WriteAlbumTit()

bool WriteAlbumTit(wxString NewAlbumTit**)** This writes the new given album title and returns true if successful.
Parameters

NewAlbumTit This is the new album title to be written to the tag.

WriteYear()

bool WriteYear(wxString NewYear**)** This writes the new given year and returns true if successful.
Parameters

NewYear This is the new year to be written to the tag.

WriteComment()

bool WriteComment(wxString NewComment) This writes the new given comment and returns true if successful.
Parameters

NewComment This is the new comment to be written to the tag.

WriteTrackNo()

bool WriteTrackNo(int NewTrackNo) This writes the new given track number and returns true if successful.
Parameters

NewTrackNo This is the new track number to be written to the tag.

WriteGenre()

bool WriteGenre(int NewGenre) This writes the new given comment and returns true if successful.
Parameters

NewGenre This is the new genre to be written to the tag.

RemoveTag()

bool RemoveTag() This completely removes the tag and returns true on success.

4.5.6 ID3v2

ID3v1 deals with all the File I/O for the ID3v2 Tags.
Note: ID3v1 & ID3v2 are subclasses of AudioFile

ID3v2()

ID3v2(wxString GivenFileName**, wxString** GivenDir**)** This is the constructor, it takes two arguments.
Parameters

GivenFileName This is the file name of the audio file.

GivenDir This is the full directory for the file.

UpdateTag()

void UpdateTag(ID3_FrameID TagFid**,char** *TagData**)** This writes the new given text to the given Field.
Parameters

TagFid this is the Fid of the tag field to change see section 8.1 on page 64 for a list of different frames.

TagData This is a C style string that contains the new data to write to the tag.

wxString GetInfo()

wxString GetInfo(ID3_FrameID TagFid**)** This returns a wxString containing the text in the given field.
Parameters

TagFid this is the Fid of the tag field to change see section 8.1 on page 64 for a list of different frames.

Chapter 5

History of Development

5.1 Language Selection & learning

5.1.1 Choosing a language

Java is the language that is most commonly used for platform independent software. However due to its nature, a Java program is generally slower than a fully compiled binary. Java is also good for GUI tools as there is a whole suite of built in libraries and facilities for GUI development.

C and C++ are another possible choice in that they have better performance than Java as a fully compiled binary. Both languages in their basic form are platform independent however there are no GUI libraries built into the standard C and C++. There are a large number of third party libraries that can be used which have a wide range of functionality including GUI development. Not all these libraries are available on all platforms. To get round the problem with differences in libraries there can be multiple copies of some code files for different platforms with classes (that may have different versions) using the same function calls.

Delphi has been available for Linux for a few years now (since 2001) in the form of Kylix a RAD (Rapid Application Development) tool provided by Borland. However this is covered by licencing laws and I would prefer to (at least primarily) develop my application using GNU (or similarly free/Open Source software).

5.1.2 finding the library

While researching libraries that I could use I did look into GTK which is available primarily for Linux and has been ported to windows (used by the GIMP successfully)

Fortunately after researching the different possibilities I found a library called WxWidgets (formally called WxWindows but had to change its name for legal reasons where another company had rights over the name, (possibly a double glazing firm) which is good for GUI programming with a number of common functions and dialogues. This library had a lot of features and resources within it for controlling the GUI and for other areas that are not normally platform independent such as file reading and writing functions.

5.1.3 WxWidgets installation problems

Previously I had never tried to install and use a library outside of the standard features in a programming language so this was a whole new area for me. I attempted to install the library on both Microsoft Windowsand Linux. I was completely unsuccessful with Microsoft Windows.

The distribution of Linux that I am using to develop the application is Libranet which is 100% Debian compliant and their for I was able to install the WxWidgets (previously WxWindows) using apt-get. however there was a problem with the dependencies in apt-get for the WxWidgets library and not everything that was required was installed. this caused a variety of unusual compilation errors that were solved when the rest of the libraries were installed.

5.1.4 learning C++ and WxWidgets

There are a small number of tutorials available for WxWidgets and a large number of examples with the distribution.

While I had a small background in the basics of C and C++ there was a lot that I had to learn (and remember after having not used either language for over a year) and so I delved into [Liberty, 2002] to get better understanding of C++.

For the learning of the WxWidgets library I used tutorials found on [Smart, 2005] which covered introductory programmes such as [Roebling, 2004] a hello world tutorial to get people started and [Beech, 1999] which provided a more in depth tutorial on the basics of WxWidgets. also available was [Smart, 2003] which is the Manual for WxWidgets providing a full list of all the classes and functions with short descriptions on how to use them.

I was able to work my way through some of the tutorials and sample programmes and with the help of [Liberty, 2002] got a good understanding of the language in two weeks. I had a testing application running which consisted of a single file borrowing heavily from one of the sample programmes in the WxWidgets package.

Development Methods

With a large project like this it is important to have an effective design strategy as a project that starts out with a loose idea and code without a plan can quickly become restrictive and problems can quickly develop low down.

5.2 Development Strategy

5.2.1 Cathedral or Bazaar

In [Raymond, 2000, The Cathedral and the Bazaa] E. S. Raymond plotted out a number of ideas and pointers for software development taken from research into similar papers and his own experience. The paper primarily compares two different methods of software development:

Cathedral This is the tried and tested 'old' method of primarily one person (or a small group) working on a project and then making releases only when most bugs have been ironed out and it is 'ready'.

Bazaar This was the name given to Linus's development strategy. 'release early and often, delegate everything you can, be open to the point of promiscuity' [Raymond, 2000]

Whilst I can't use a bazaar method of development due to the nature of this project being a university one which has to be my own work I feel the paper covers a number of points that are worth considering in the whole scheme of program development. Below I have lists a number of points extracted from the paper that I feel are relevant to the development of my project:

Pointers from the Bazaar

1 Every good work of software starts by scratching a developer's personal itch.

2 Good programmers know what to write. Great ones know what to rewrite (and reuse).

3 "Plan to throw one away; you will, anyhow." (Fred Brooks, The Mythical Man-Month, Chapter 11)

7 Release early. Release often. And listen to your customers.

9 Smart data structures and dumb code works a lot better than the other way around.

11 The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.

18 To solve an interesting problem, start by finding a problem that is interesting to you.

Use and Interpretation of pointers

1 I personally have a large number of digital audio files some of which are well organised others aren't.

2 While I aim to write most of the code of this project there is little need in 'Reinventing the wheel' so libraries can save time as well as small sections of code from open source projects can reduce development time.

3 While I aim for my project to be designed well it is clear that even the best hackers have to rewrite sections of code in projects so I should be prepared to spend some time updating and redeveloping some of my program structure.

7 Whilst I do not aim to widely distribute my software I do intend (once I have a stable program that has limited functionality) to get people to test the application and give me feedback.

9 while I aim to have as little as possible 'Dumb code' I am aware of the importance of good data structures, and also this rule applies to the construction of objects so that updates don't cause huge problems throughout the program.

11 Similar to 7 I am hoping to get some useful feedback from people when they try my application.

18 This is similar to the first issue.

5.2.2 Influence of now ideas & resources

Due to using many things that are new to me including the GUI Development and other libraries and the complexity of the problem being bigger than anything I have as yet taken on I have to do a lot of testing and prototyping alongside the general implementation of features. because of this I have to have a system of:

prototype → evaluate → implement → evaluate → improve

And because of this development procedure I have to design my program to be flexible and have good separation of functionality from class to class.

5.3 Development & initial prototype

5.3.1 Makefile

When developing and testing an application it often requires to be compile many times (hundreds if not thousands over the period of a few months). this is not such a problem for a small C or C++ program where the command line to compile it may be

```
gcc myfile.c -o myprog
```

however when you are using a number of libraries and a number of files the command for compile and link a program can be much larger and typing out a 20+ character command is not much fun, this is made easier by the means of a makefile. I got details on this from [Sobell, 1997].

this allows me to build my application with the command:

```
make
```

See section 8.3.2 on page 70 for my full makefile.

5.3.2 Minus number

for a short while I was having problems getting the TAG information from files (the last 128KB of the file) I was getting random characters and unrecognised symbols. I finely found out that I was reading 128KB after the end of the file then starting from there rather than going 128KB BEFORE the end of the file and reading from there.

the code change was simple, this code:

```
TheFile.SeekEnd(128);
```

was changed to:

```
TheFile.SeekEnd(-128);
```

5.3.3 List

One major function of the application is to list all the audio files so that they can be easily selected and edited. Fortunately There are several types of list in the WxWidgets library. The basic list does not support more than 3000 items in a list, considering that my MP3 collection (all be it unusually large) has 7,000+ items I needed something that could handle this many items. so I went for the far more complex list option "wxListCtrl" which is documented in [Smart, 2003].

This list as well as being another control from the library is a very complex one with many different viewing modes and events that can be tracked. it is a good example of how fully featured the library is as everything you could possibly want to do with a list is in here. it even includes a very basic search facility. It took me a couple of weeks to get the list working fully due to its complexity.

5.3.4 Good Code / Bad Code

The testing code I did was a complete mess and made almost no use of object oriented design outside of its use of the WxWidgets library. So I completely stripped down my code and restructured it so that I had a separate “AudioFile” object that was used to control the mp3 files. I also separated the classes into their own separate .cpp and .h files for my personal ease of navigation as the file was getting larger at 700 lines including lots of messy comments. This took some time but was worth it as it indirectly increased the functionality and whilst at the time it was not a major issue the performance increased a small amount.

Got the loop working

As part of the conversion to more object oriented code I managed to fix what was a large bug. I had not constructed the loop calling procedure. this meant you could hi light several items from the list and only one of them would be picked to have the selected ”action” performed. the code has now been changed and commented so that this now works.

5.3.5 Searching

The area of searching and in particularly text matching is an enormous subject on its own.

After lots or reading and furthering my knowledge of the wxListCtrl I found that it has a built in search facility, however this facility is limited. the search is case sensitive and does not allow sub string searching. so I have found a number of library functions which are part of the wxString class. They allow for comparing strings in a number of different ways.

Due to the complexity of the subject area I have started off small with as much room for easy expansion as I can make. Rather than defining a search within the function that is called when a user selects to search for something I have made a separate function (may become a class) for searching which will allow for the search to be made more complex.

5.3.6 Running applications

Normally launching a third party application is different from platform to platform. In Linux you have to use the system calls ”Fork” and ”exec” however this is a Unix system call and would not work under other operating systems. fortunately WxWidgets has a library function for this. Allowing for an application to be easily launched within any Operating system. see section 2.7 on page 20 for more information on this issue.

5.3.7 Vectors

the list has been remodeled to control a Vector, this allows for an indefinite number of AudioFile objects to be stored. the objects are created and then given pointers. Using the vector allows for the objects to be stored in memory and not have to be recreated every time (this has made the running of the program noticeably faster due to the AudioFile objects not having to be continually recreated and thus reducing the File I/O time).

I had an amount of difficulty with the vectors as I'd never used them before. the problem I'd had was that I was trying to put the objects directly into the vector rather than just using pointers to objects in the vector.

5.3.8 Options File

An ”options” file has been created called ”mp3org.conf”. it is a text file which gets read on program loading by an options class. The idea is that the options class can be expanded to hold more centralised options and settings for the program. it currently only holds the Media player application associated with the application and the file browser.

Initial Solution

The initial file contains a number of loops and works with character comparisons. Whilst it works I'm not too happy with the way it works and hope to be able to clean up the class and make it friendlier to use and update and remove some of the messy loops.

Lua

One way I have found to possibly clean up the options file handling is to use a language called Lua see section 2.2 on page 10 for full details. Lua has the ability to read a text file (that looks similar to a Windows.INI file as its input for assigning variables. If I can get this to work then it will fully manage The options for me (including the File I/O) allowing for a clear structure in the options file with comments and also some Lua code can be used if required to modify settings. Sadly I have been unable to get Lua working with the application working

5.3.9 Genres

One aspect of an ID3v1 tag is that it can contain a genre. this is stored as an integer in 1 byte of the file. First of all this number has to be converted to a string that is meaningfull to a user as a user will probably not realise that Genre 0 is Blues etc...

Borrowed code

Fortunately I was able to take some code from an existing project and modify it for my own use (partly because it saved me typing out over 140 genres). The program I borrowed the code from was [myers carpenter, 2005] a small command line tag editing program.

Implementation

This I was able to combine with a list box so users were able to change the genre by selecting another one from the menu. I did also find a small issue in how the information was read from the file. any number over 128 was coming back as a minus number due to the fat it was being seen as a signed int so I changed the file reading buffer to an 'unsigned char' and this solved the problem.

Christian Gangsta Rap

on a small side note I still find it hard to believe that there is a need for the genre 'Christian Gangsta Rap' (which is defined by the ID3v1 standard) and a google search found this little ditty.

> He's the Jizzle to the E, Sizzle U Sizzle
>
> All my boys in the hizzle better feel His love
>
> Mo'fuckers try to haterise, say he ain't no Messiah
>
> But I whip out my glock and send em to Hell fire.
>
> Sons of God in the House, wave your motherfuckin hands
>
> Don't give the fucking pigs a second damn glance
>
> Smoke a blunt pass it left, at His left sat the Christ.
>
> Drink gin and juice til you puke like a fuckin Poltergeist.
>
> Word to your mother.

`<http://interrobang.jwgh.org/songs/index.php?song=Christian%20Gangsta`

5.4 Extension of features

5.4.1 Context Menu

I wanted a menu to appear when you right clicked on the list object, I managed to get everything working except for the right click event, I could make the menu appear from other events such as selecting an item from a menu. The solution was changing the event ID that I had given the right click to wxID_ANY for some reason it did not like me setting it an event id myself.

5.4.2 MSW compile

Finally I have managed to get the program to compile under Microsoft Windows. I have had to make a few subtle changes to code so it compiles under windows. The windows executable looks ugly and some functionality doesn't work fully so I will have to do some further development to the code under windows to make some features work but that will mainly be small changes... hopefully.

5.4.3 Play lists

I have implemented the functionality to write to a m3u play list for details on the play list file format see section 2.5.1 on page 17. This is built into a popup menu covered in more detail in section 5.4.4 on page 56.

5.4.4 Popup Menu

This is a menu that appears when a user right clicks on an item in the list of files. I had some difficulty with the implementation of this feature due to the way in which you use enumerations with WxWidgets (and how this does not work if you set it an enumerated value). I found i was able to launch (and use) the menu from other events which alerted me to the fact that the problem with the menu was in the event handeling.

5.4.5 Sorting Files

Despite this being one of the core objectives of the project (and majour aspects of functionality) I had got the basic functionality (from a code perspective) covered in other features of the programme and so this was left to a later date to be implemeanted whilst i payed more attention to more complex aspects of the application functionality. However it was finaly implemeanted and added into the menu system.

5.5 Insertion & Removal of Preceding Track Numbers

This is an aspect of functionality that came from user feedback after some testing of a working application with basic functionality. Sometimes files are named with the track numbers at the start of the file name. Some users asked if it were possible to quickly rename (otherwise correctly taged) files with preceding track numbers so that files would be listed in directories in album order. Similarly some users complained of tracks being named with preceding Track numbers and wanted to be able to remove them quickly. Because of this being a generaly easy aspect of functionality to implemeant I was able to add both features to a dropdown menu.

Chapter 6

Discussion & evaluation

For my Evaluation i'm going to compare the project at this stage of developmeant with my original specification and evaluate how well it meets the the specification. I'm also going to include a short comparison with some similar applications. Then i'm going to look into areas that need further developmeant or functionality that is missing. In the discussion i'm going to look into areas that were researched and not implemeanted, and look into why things weren't implemeanted. Finaly i'll look into the pros and cons of the ideas that i used.

6.1 Comparison with specification

Here I am comparing my application with the specification i set myself. i'm going to do this by listing the pointers from my specification the writing about how well I met that aspect of the specification.

Edit/create tags

My application Works with code i have written for ID3v1 and with the ID3lib library for ID3v2. I would like to further my support for ID3v2 but i am happy with the level of support that i have got.

Multiple renaming

Using the user interface code i am able to apply the same change to all selected files, This is an area of code that i would like to tidy up as i feel that parts of it are a 'bad hack'. However as a programme it works fine. I would like to tidy up the user Interface part of this a little.

File renaming

This i managed to do in a number of different ways including the adding and removing of preceding track numbers see section 5.5 on page 57. I would like to extend this to give a user more controle over what to name the file.

File copying

As far as any level of evaluation this functionality can go, it can eyther work or not work. Obviously i got this working, This was a simple addition using similar code to the file renaming.

Play files

I got this functionality working and also added the facility to launch a chosen file browser with similar code.

Sort files

This functionality was added towards the end of developmeant and it fulfils my basic requirmeant of sorting a file into directories (and renaiming the file) with practical orderly names extracted from the extended information.

Search feature

This functionality is in the programme but due to time constraints it only has a basic level of functionality (pattern matching) and no advanced algorithms for inclusion of full stops or fuzzy matching for spelling mistakes.

6.2 Comparisons with existing applications

Below I have compared my application with a number of similar applications (that i looked at in my research) to compare functionality and usability. For more information on thease applications look at section 2.9 on page 23.

6.2.1 EasyTAG

My application is not as fully featured as EasyTAG, however my application is platform independent and its simple user interface makes the functionality that it does have fast to perform and a user can use it under any operating system they choose. Also my application can be easily extended to support the other file types supported by EasyTAG. perhaps after another years development my application will be closer to the functionality of EasyTAG

6.2.2 mBox

mBox Does have a slight different angle of main functionality to my programme in that mBox is designed primarily to convert from one digital audio file format to another and part of its functionality is to edit the tags on the new/existing files. However, the level of functionality is similar for tag editing as the same areas are supported. My program is easier to use due to it being designed specifically for the task of editing tags.

6.2.3 Cantus

Cantus is one of the big names in this area of application due to it being developed by a very experienced developer, The about box in the application comments that the amount of development time totals over 400 hours from the primary developer (the total is almost impossible to calculate due to this being an open source project). My programme has one advantage over Cantus in that for smaller functions my programme is simpler and faster to use (by a small margin) however my programme due to its object oriented design can be easily modified to obtain more functionality with a comparatively small development time overhead.

6.3 Areas for further developmeant

Whilst the application is in a fully working and usefull stage it is verry rare that an application gets to a stage where nothing can be added (or indeed taken away if you read the Cathedral & the bazzaar [Raymond, 2000]) And on thease grounds i can think of a number of changes that need to be made to my application to update it.

Chapter 7

Conclusion

The Project has lasted from October 2004 - April 2005 and has proved to be both interesting and educational to me. I have achieved my aim of creating a GUI application for editing the extended information within audio files and allowing the functionality to sort these files into sub directories based on album and artist information. The application does both of these with a good variety of functionality. due to its object oriented design the application is both easily maintainable and some of the classes are potentially easy to copy into other similar applications.

My project works entirely to my specification and has taken me a lot of time. on those grounds give me a first (plus a bit extra to save me from making a mess of my other three modules)

are there any marks for having over 2000 lines of code?

Bibliography

[Abels, 2005] Abels, S. (2005). *cantus 1.07*. `<http://freshmeat.net/projects/cantus>`. The website of the cantus project.

[Beech, 1999] Beech, D. (1999). *WxWidgets tutorial*. `<http://www.bzzt.net/~wxwindows/icpp_wx1.html>`. An intuitive tutorial on using many features of WxWidgets.

[Cadenhead, 2001] Cadenhead, R. (2001). *Sams Teach Yourself Java in 24 Hours*. Sams, 2nd edition. only used for Java code on how to read from ID3v1 tags.

[easyTAG team, 2005] easyTAG team (2005). *EasyTAG*. `<http://easytag.sourceforge.net/>`. The website of the EasyTAG project.

[Gonze, 2003] Gonze, L. (2003). *A survey of playlist formats*. `<http://gonze.com/playlists/playlist-format-survey.html>`. A collection of information documenting different playlist file types.

[Hnilica, 2005] Hnilica, M. (2005). *mBox*. `<http://www.mbox.wz.cz/>`. The website of the mBox project.

[Langen, 2005] Langen, C. (2005). *wxmusik*. `<http://musik.berlios.de/>`. The website of the wxmusik project.

[Liberty, 2002] Liberty, J. (2002). *Sams Teach Yourself C++ in 24 Hours*. Sams, 3rd edition.

[Lua-website, 2005] Lua-website (2005). *Lua Website*. `<http://www.lua.org/>`. The Lua project website.

[Mahoney, 2005] Mahoney, Dirk, e. a. (2005). *id3lib - The ID3v1/ID3v2 Tagging Library*. `<id3lib - The ID3v1/ID3v2 Tagging Library>`. Website containing the library id3lib used for reading ID3v2 Tags.

[myers carpenter, 2005] myers carpenter (2005). *id3v2*. `<http://id3v2.sourceforge.net/>`. A small application for editing ID3v2 tags.

[Nilsson, 2004] Nilsson, M. (2004). *ID3v1*. `<http://www.id3.org/id3v1.html>`. Website documenting how the ID3 Tags work on music files (ID3v1, ID3v1.1, ID3v2).

[ogg-vorbis team, 2005] ogg-vorbis team (2005). *Vorbis.com*. `<http://www.vorbis.com/>`. The website of the Ogg Vorbis group containing information on ogg compression.

[Raymond, 2000] Raymond, E. S. (2000). *The Cathedral and the Bazaar*. `<http://www.catb.org/˜esr/writings/cathedral-bazaar/>`. A paper written about different developmeant techniques.

[Roebling, 2004] Roebling, R. (2004). *hello world tutorial*. `<http://wxwindows.org/hello.htm>`. hello world tutorial for beggining using WxWidgets.

[Smart, 2005] Smart, Julian, e. a. (2005). *WxWidgets Website*. WxWidgets, `<http://www.wxwidgets.org/>`. The homepage of the wxWidgets project.

[Smart, 2003] Smart, J. (2003). *WxWidgets Manuel*. WxWidgets, `<http://wxwindows.org/manuals/2.4.2/wx.htm>`. A full guide written in HTML containing details of all the classes and functions and their uses.

[Sobell, 1997] Sobell, M. G. (1997). *A Practical Guide to Linux*. Addison Wesley, 1st edition. A little out of date but usefull for information on linux and Makefiles.

[unknown, 2005] unknown (2005). *KIDiddles: Song Lyrics: Ten Green Bottles*. `<http://www.kididdles.com/mouseum/t050.html>`. Lyrics to 'Ten Green Bottles' on the KIDiddles website.

Chapter 8

Appendices

8.1 ID3v2 Frames

Below is a complete list of all the ID3v2 Frames.

```
Audio encryption
Attached picture
Comments
Commercial frame
Encryption method registration
Equalization
Event timing codes
General encapsulated object
Group identification registration
Involved people list
Linked information
Music CD identifier
MPEG location lookup table
Ownership frame
Private frame
Play counter
Popularimeter
Position synchronisation frame
Recommended buffer size
Relative volume adjustment
Reverb
Synchronized lyric/text
Synchronized tempo codes
Album/Movie/Show title
BPM (beats per minute)
```

```
Composer
Content type
Copyright message
Date
Playlist delay
Encoded by
Lyricist/Text writer
File type
Time
Content group description
Title/songname/content description
Subtitle/Description refinement
Initial key
Language(s)
Length
Media type
Original album/movie/show title
Original filename
Original lyricist(s)/text writer(s)
Original artist(s)/performer(s)
Original release year
File owner/licensee
Lead performer(s)/Soloist(s)
Band/orchestra/accompaniment
Conductor/performer refinement
Interpreted, remixed, or otherwise modified by
Part of a set
Publisher
Track number/Position in set
Recording dates
Internet radio station name
Internet radio station owner
Size
ISRC (international standard recording code)
Software/Hardware and settings used for encoding
User defined text information
Year
Unique file identifier
Terms of use
Unsynchronized lyric/text transcription
Commercial information
Copyright/Legal infromation
Official audio file webpage
Official artist/performer webpage
Official audio source webpage
Official internet radio station homepage
Payment
Official publisher webpage
User defined URL link
```

8.2 User Guide

This is a guide on how to use the features of mp3org. It covers all the basic functionality plus some extra features.

Load application

Start Programme

The application is launched by running the binary file ("mp3org.exe" under windows or "mp3org" under Linux). when the application loads it will read from the current working directory the configuration file (If not found default options are used).

List Files

Before you can do anything with any files you have to list all the files in the directory you wish to look at. This includes reading all the existing extended information on those files. this is done by going to the menu and selecting:

`File>Dir`

Then using the Directory selection box you select the directory that you want to look at the files in. and click OK. then the files will be listed in the list box on the main window of the application and some of their Tag data will be shown as well.

Edit ID3v1 Tag

Once you have got a list of file(s) in a directory you can highlight one (or more) and select from the drop down menu:

`ID3v1>Edit Tag`

This can also be done by right clicking on the item(s) in the list and selecting "Edit ID3v1 Tag"

Then the ID3v1 Edit dialog will be shown with the first selected file's ID3v1 tag information. Once you have made your changes to this file you can click "OK" to save them or “Cancel” to not save any changes. after clicking “OK” or “Cancel” the next selected files’ tag is displayed for editing till the last tag has been edited.

Edit ID3v2 Tag

Once you have got a list of file(s) in a directory you can highlight one (or more) and select from the drop down menu:

```
ID3v2>Edit Tag
```

This can also be done by right clicking on the item(s) in the list and selecting “Edit ID3v2 Tag”

Then the ID3v1 Edit dialog will be shown with the first selected file’s ID3v1 tag information. Once you have made your changes to this file you can click “OK” to save them or “Cancel” to not save any changes. after clicking “OK” or “Cancel” the next selected files’ tag is displayed for editing till the last tag has been edited.

Batch Edit ID3v1 Tag

To edit multiple tags simultaneously (eg. set all files to the same artist) select the desired files to edit in the same way as before and select from the drop down menu:

```
ID3v1>Batch Edit Tag
```

This opens up a dialog similar to before except that this time no information is displayed and any fields you add text to will overwrite all the same fields in the file (eg. if you change the contents of the artist box all the files will have their artist updated and whatever was there before will be replaced). any fields left blank will not be altered

Again you can click “OK” to perform the changes or “Cancel” to not make any changes. there is only one dialog here that changes all selected files.

Copy out files

Once you have finished editing tags with the application, you can then select all the files you want to sort and the program will copy them to a location of your choice with the directory and file name structure:

```
/<BAND NAME>/<ALBUM NAME>/<TRACK NUMBER> - <TRACK NAME>.mp3
```

To do this select the files as normal and select from the drop down menu:

```
FileI\O>Sort Files
```

Play files

To play selected files in a media player (selected by you in the configuration file) select the files as normal and select from the drop down menu:

`Third Party Apps>Play Selected Files`

Add To Play List

There is a facility to add files to a m3u play list (the play list has to be referenced in the configuration file). to do this select the files as normal and select from the menu that appears when you right click on the list:

`Add to play list>NAME OF PLAYLIST`

Note that the name of the play lists will be shown and you click on the play list you want to add the file to.

Rename with/without preceding track name

To add or remove the track number to files. this is done by selecting the files as normal and selecting from the menu (Respectively):

`File I\O>Rename With Track No.` ← To add the track number

`File I\O>Rename With Track No. Removed` ← To remove the track number

This is similar to sorting files, it allows you to copy files. the files are copied rather than being edited directly allows the user to check the new files to save original files being edited wrongly or corrupted.

Search

This provides the user with a facility to select files in the list with pattern matching. Select search from the file menu.

`File>Search`

A small dialog appears allowing you to type in the text you wish to search for, on pressing OK the list is searched and any matches become highlighted (selected) and actions can be performed on the matches.

Configuration File

The application looks for a file named ‘mp3org.conf’ atruntime (it looks for this file inthe current working directorie) and this file is a text file with various options in it shown below. If the file is not found (or one of the important options is not set) then there are defult options built into the application. Below is a sample file:

```
#lines starting with hash are ignored
#all items must be on their own line with NO uneeded spaces
#remember applications are case sensative!

#third part applications
MediaPlayer:xmms
FileBrowser:nautilus --no-desktop --browser
#programme settings
ProgrammeWidth:640
ProgrammeHeight:480

#play lists
PlaylistFile:test2.m3u
PlaylistDir:/home/amon/Documents/Uni Work/FYP/mp3org
PlaylistFile:test.m3u
PlaylistDir:/home/amon/Documents/Uni Work/FYP/mp3org
#there must be a new line at the end of the file
```

8.3 code

8.3.1 ID3v1 reading code

I was able to get a good head start by getting the following Java code from a book [Cadenhead, 2001] where there was an example of reading strings from files that used MP3 files:

```
import java.io.*;

public class ReadID3
{
 public static void main(String[] arguments)
 {
  try
  {
   File song = new File(arguments[0]);
   FileInputStream file = new FileInputStream(song);
   int size = (int)song.length();
   file.skip(size - 128);
   byte[] last128 = new byte[128];
   file.read(last128);
   String id3 = new String(last128);
   String tag = id3.substring(0, 3);
   if (tag.equals("TAG"))
   {
    System.out.println("Title:" + id3.substring(3, 32));
    System.out.println("Artist:" + id3.substring(33, 62));
    System.out.println("Album:" + id3.substring(63, 91));
    System.out.println("Year:" + id3.substring(93, 97));
    System.out.println("Comment:" + id3.substring(98, 127));
    System.out.println("genre:" + id3.substring(128));
   }
   else
   System.out.println(arguments[0] +" does not contain ID3 info.");
   file.close();
  }\documentclass[12pt,a4paper,english]{report}
   catch (Exception e)
  {
   System.out.println("Error --" + e.toString());
  }
 }
}
```

8.3.2 Makefile

This is my Makefile for the program.

```
Makefile
```

8.3.3 My Full Code

From here Follows my entire code listing.