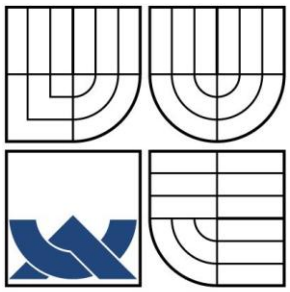
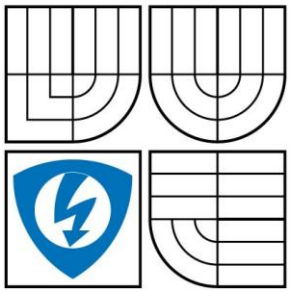


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

DATABÁZE ZVUKOVÝCH SOUBORŮ

DATABASE OF AUDIO RECORDS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC.NABHAN KHATIB
AUTHOR

VEDOUCÍ PRÁCE — ING. IVAN MÍČA
SUPERVISOR

BRNO 2009

Databáze zvukových souborů

Vytvorte aplikaci v jazyce C++ s grafickým uživatelským rozhraním, která bude schopná komunikovat se vzdáleným serverem a provádet základní funkce: napr. nahrávání a stahování souboru. Aplikace Bude používat databázový servere SQL. Aplikace navíc musí provádet nekteré vybrané metody zpracování signálu, jako jsou napríklad komprese ci dekomprese ruznými kompresními algoritmy, prevzorkování,
filtrace a další. Aplikace by mela umožnovat prístup uživatelum do databáze pomocí uživatelského jména a hesla a adminitrátorovi by mela umožnit pridelovat ruzná oprávnení pro jednotlivé uživatele.

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Ivanu Míčovi, doktorandovi Ústavu telekomunikací na FEKT VUT v Brně, za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne ....

Podpis autora

Prohlášení

Prohlašuji, že svou diplomovou práce na téma DATABÁZE ZVUKOVÝCH SOUBORŮ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… ……..

podpis autora

KHATIB, N. Databáze zvukových souborů. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 44 s. Vedoucí diplomové práce Ing. Ivan Míča.

Contents

Introduction

Programming language has a loose definition, it used by a programmer to give the machine very specific instructions in order to serve some purpose for the user.
programming language is designed to express computation that can be performed by a machine, particularly a computer. Programming languages can be used to create programs that specify the behavior of a machine, to express algorithms precisely, or as a mode of human communication.
A programming language provides a structured mechanism for defining pieces of data, and the operations or transformations that may be carried out automatically on that data. The abstractions present in the language allow a programmer to represent the concepts involved in a computation as a collection of the simple elements available
(called primitives).
Programming languages (sometimes also known as computer languages differ from most other forms of human expression in that they require a greater degree of precision and completeness. When using a natural language to communicate with other people, human authors and speakers can be ambiguous and make small errors, and still expect their intent to be understood. However, figuratively speaking, computers "do exactly what they are told to do", and cannot "understand" what code the programmer intended to write. The combination of the language definition, a program, and the program's inputs must fully specify the external behavior that occurs when the program is executed, within the domain of control of that program.
The C API is a powerful code distributed with MySQL. It is included in the mysqlclient library and allows C programs to access a database.
Mysql provides a client library written in the C programming language that you can use to write client programs that access MySQL databases.
SoX is a cross-platform (Windows, Linux, MacOS X, etc.) command line utility that can convert various formats of computer audio files in to other formats. It can also apply various effects to these sound files, and, as an added bonus, SoX can play and record audio files on most platforms.

1 High-level and Low-level programming language

There is a difference between High level and low level Programming Languages, so a high-level programming language is a programming language that, in comparison to low level programming language may be more abstract, easier to use, or more portable across platforms, also that they are easier to read, write, and maintain. Ultimately, programs written in a high-level language must be translated into machine language by a compiler or interpreter .

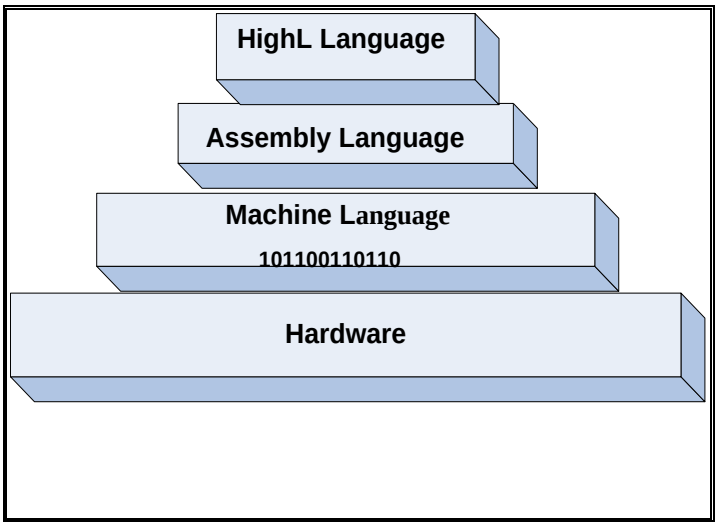


Figure1.1 Programming language.

A programming langauge such as C that enables a programmer to write programs that are more or less independent of a particular type of computer. Such languages are considered high-level because they are closer to human languages and further from machine languages. In contrast, assembly languages are considered low-level because they are very close to machine languages.

2 C++

C++ is a general purpose programming language that is considered a mid-level language due to its leniency towards both high-level and low-level functionalities. In 1979, an employee of Bell Labs, Dr. Bjarne Stroustrup, developed C++ as an improvement on the existing C programming language, a result of an idea he had while programming for his Ph.D. thesis. Dr. Bjarne Stroustrup originally called his enhancement "C with classes"; it was renamed to C++ in 1983.

The standards used by C++ were officially sanctioned as ISO/IEC 14882:1998, of which 2003 is the most current. An underground version of the standard is being developed to replace ISO/IEC 14882, which is informally known as C++0x. C++ is used worldwide by computer programming students creating their "Hello, World" programs and coding gurus creating the most sophisticated software for FORTUNE 500 businesses.

C++ is a great choice for so many different purposes because it supports various programming styles, it avoids having features that make it platform dependent or that are not general purpose, and it does not incur overhead for features that are not used (resulting in slimmer and faster programs).

However, there have been problems surrounding the use of C++ since its birth. Until recently, making a C++ compiler version that is standards compliant was proving very difficult because the different compilers used different levels of compliance with the standard. Post 1998 releases of popular C++ compilers have conformed to support the 1998 C++ standard, eliminating most of that problem. Other critics raise the point that C is its parent language; therefore C++ inherits the remarks targeted at its progenitor. One thing is for certain, for being such a heavy hitter for such a new appearance, C++ is going to be around for a long time to come.

2.1 C++ Builder

The c++ launguage is at the core of C++ Builder and it is a rapid application development (RAD) environment produced by the CodeGear subsidiary of Borland for writing programs in the C++ programming language. C++ Builder combines the Visual Component Library and IDE written in Delphi with a C++ compiler. The release cycle is such that Delphi gets major enhancements first, with C++ Builder following.[1] Most components developed in Delphi can be used in C++ Builder with no modification, although the reverse is not true.

C++ Builder includes tools that allow true drag-and-drop visual development, making programming easier by incorporating a WYSIWYG GUI builder into its IDE.

C++ Builder can be used to create 32-bit Windows applications ranging from general-purpose utilities to sophisticated data access programs. When you start C++ Builder, you are placed within the object-orientated integrated development environment, the IDE. This environment provides all the tools you need to design, develop, test, debug, and deploy applications.

2.2 Header file

In computer programming, particularly in the C and C++ programming languages, a header file or include file is a file, usually in the form of source code, that is automatically included in another source file by the compiler. Typically, header files are included via compiler directives at the beginning (or head) of the other source file.

In C++, the Standard Library is a collection of classes and functions, which are written in the core language. The Standard Library provides several generic containers, functions to utilise and manipulate these containers, function objects, generic strings and streams (including interactive and file I/O), support for some language features, and every day functions for tasks such as finding the square root of a number

A header file commonly contains forward declarations of classes, subroutines, variables, and other identifiers. Identifiers that need to be declared in more than one source file can be placed in one header file, which is then included whenever its contents are required.

In the C and C++ programming languages, standard library functions are traditionally declared in header files.

The Standard C++ Library is composed of eight special-purpose libraries:

- The Language Support Library
- The Diagnostics Library
- The General Utilities Library
- The Standard String Templates
- Localization Classes and Templates
- The Containers, Iterators and Algorithms Libraries (the Standard Template Library)
- The Standard Numerics Library
- The Standard Input/Output Library
- C++ Headers for the Standard C Library

2.2.1 windows.h

Windows.h is a header file for the C programming language which contains declarations for all the function in the Windows API, all the common macros used by Windows programmers, and all the data types used by the various functions and subsystems. It defines a very large number of Windows specific functions that can be used in C. The Win32 API can be added to a C programming project by including the <windows.h> header file and linking to the appropriate libraries.

2.2.2 mysql.h

mysql.h defines the primary MySQL-related constants and data structures.

3 SSL (Secure Socket Layer)

SSL (Secure Socket Layer) is a kind of protocol for encrypting and decrypting data sent across direct internet connections. When a client makes an SSL connection with any kind of server, all data sent to and from that server is encoded with a complex mathematical algorithm that makes it extremely difficult to decode anything that is intercepted.

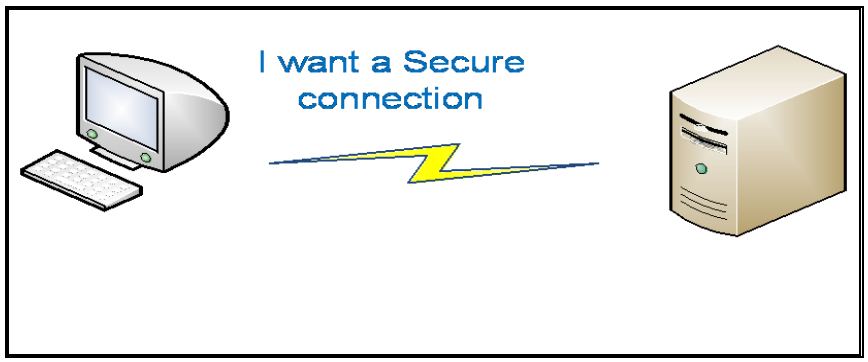


Figure 3.1 Secure Socket Layer.

- The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.
- The server sends the client the server's SSL version number, cipher settings, randomly generated data, and other information the client needs to communicate with the server over SSL. The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client's certificate.
- The client uses some of the information sent by the server to authenticate the server (see Server Authentication for details). If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established.
- Using all data generated in the handshake so far, the client (with the cooperation of the server, depending on the cipher being used) creates the premaster secret for the session, encrypts it with the server's public key, and sends the encrypted premaster secret to the server.
- If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case the client sends both the signed data and the client's own certificate to the server along with the encrypted premaster secret.
- If the server has requested client authentication, the server attempts to authenticate the client (see Client Authentication for details). If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the master secret.
- Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity--that is, to detect any changes in the data between the time it was sent and the time it is received over the SSL connection.
- The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.
- The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.
- The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

If the connection is rejected, a fail message is sent to the client. If the connection is accepted, or if the server is not set up to receive certificates, it decodes the session key from the client with its own private key and sends a success message back to the client, thereby opening a secure data channel.
Client. Any program that is able to make an SSL connection.
Session Key. The session key is what both the client and the server use to encrypt data. It is created by the client.

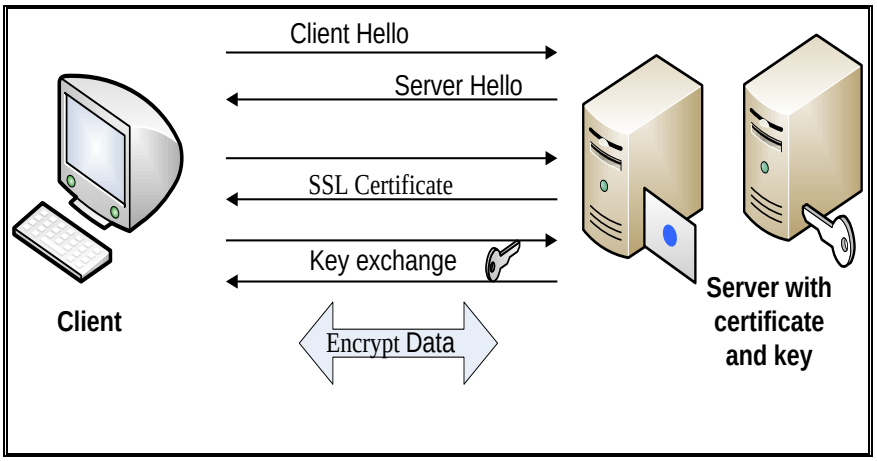


Figure 3.2 SSL Connection.

Public Key. The public key is the device with which the client encrypts a session key. It does not exist as a file, but is a by-product of the creation of a certificate and private key. Data encrypted with a public key can only be decrypted by the private key that made it.

Private Key. The private key decrypts the client's session key that is encrypted by a public key. The private key file has the .key ending. Private keys should NEVER be distributed to anyone.

Certificate: The Certificate file holds the identification information of the client or server. This file is used during connection negotiations to identify the parties involved. In some cases, the client's certificate must be `signed' by the server's certificate in order to open an SSL connection. Certificate files have the .crt ending

A digital certificate is a digital credential that validates the identity of the certificate's owner, much as a passport does. The identification information that a digital certificate provides is known as the subject distinguished name. A trusted party, called a Certificate Authority (CA), issues digital certificates to users or to organizations. The trust in the CA is the foundation of trust in the certificate as a valid credential

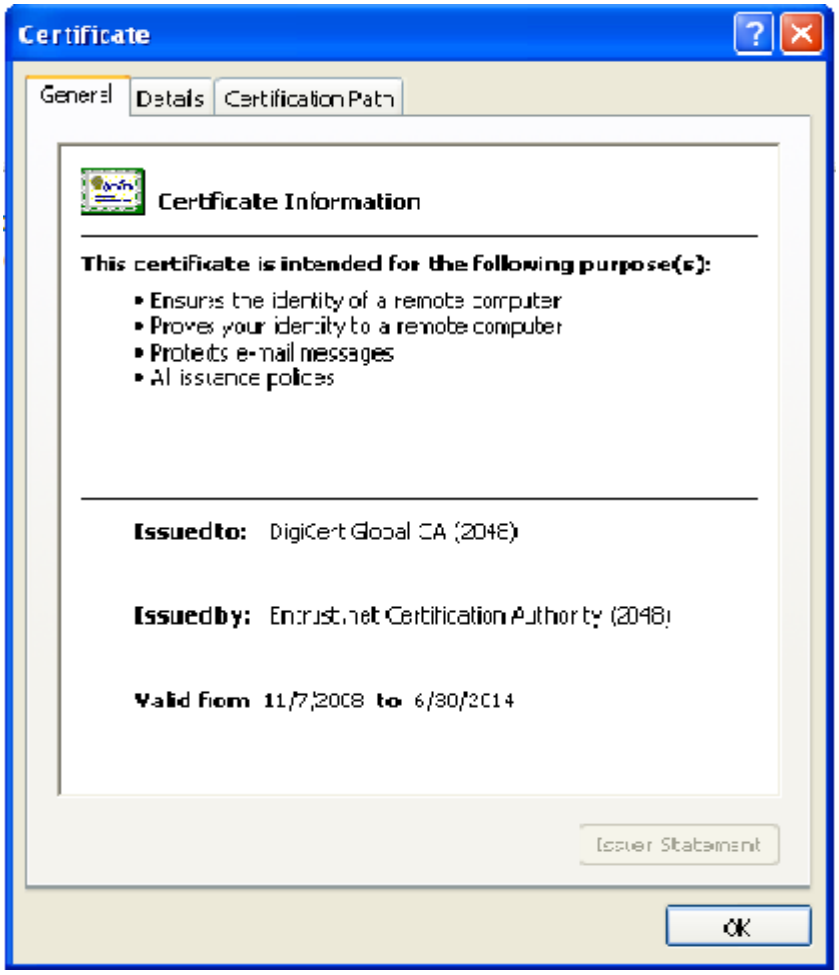


Figure 3.3 Digital certificate.

A digital certificate also contains a public key which is part of a public-private key pair. A variety of security functions rely on the use of digital certificates and their associated key pairs. You can use digital certificates to configure Secure Sockets Layer (SSL) sessions to ensure private, secure communication sessions between users and your server applications. You can extend this security by configuring many SSL-enabled applications to require certificates instead of user names and passwords for more secure user authentication

An SSL certificate may also contain the following information:

- The domain for which the certificate was issued.
- The owner of the certificate (who is also the person/entity who has the right to use the domain).
- The physical location of the owner.
- The validity dates of the certificate.
- Owner's public key
- Owner's name
- Expiration date of the public key
- Name of the issuer (the CA that issued the Digital Certificate
- Serial number of the Digital Certificate
- Digital signature of the issuer

certification authority is Abbreviated as CA, a trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The role of the CA in this process is to guarantee that the individual granted the unique certificate is, in fact, who he or she claims to be. Usually, this means that the CA has an arrangement with a financial institution, such as a credit card company, which provides it with information to confirm an individual's claimed identity. CAs are a critical component in data security and electronic commerce because they guarantee that the two parties exchanging information are really who they claim to be.

SSL can use these algorithms:

- DES. Data Encryption Standard, an encryption algorithm used by the U.S. Government.
- DSA. Digital Signature Algorithm, part of the digital authentication standard used by the U.S. Government.
- KEA. Key Exchange Algorithm, an algorithm used for key exchange by the U.S. Government.
- MD5. Message Digest algorithm developed by Rivest.
- RC2 and RC4. Rivest encryption ciphers developed for RSA Data Security.
- RSA. A public-key algorithm for both encryption and authentication. Developed by Rivest, Shamir, and Adleman.
- RSA key exchange. A key-exchange algorithm for SSL based on the RSA algorithm.
- SHA-1. Secure Hash Algorithm, a hash function used by the U.S. Government.

Key-exchange algorithms like KEA and RSA key exchange govern the way in which the server and client determine the symmetric keys they will both use during an SSL session. The most commonly used SSL cipher suites use RSA key exchange.

4 MySQL

MySQL is a relational database management system (RDBMS) based on SQL (Structured Query Language). First released in January, 1998, MySQL is now one component of parent company MySQL AB's product line of database servers and development tools.
Many Internet startups became interested in the original open source version of MySQL as an alternative to the proprietary database systems from Oracle, IBM, and Informix. MySQL is currently available under two different licensing agreements: free of charge, under the GNU General Public License (GPL) open source system or through subscription to MySQL Network for business applications.
MySQL runs on virtually all platforms, including Linux, Unix, and Windows. It is fully multi-threaded using kernel threads, and provides application program interfaces (APIs) for many programming languages, including C, C++, Eiffel, Java, Perl, PHP, Python, and Tcl.
MySQL is used in a wide range of applications, including data warehousing, e-commerce, Web databases, logging applications and distributed applications. It is also increasingly embedded in third-party software and other technologies.

5 The MySQL C API

The C API code is distributed with MySQL. It is included in the mysqlclient library and allows C programs to access a database.
MYSQL PROVIDES A CLIENT LIBRARY WRITTEN in the C programming language that you can use to write client programs that access MySQL databases.This library defines an application programming interface that includes the following facilities:
Connection management routines to establish and terminate a session with a server.
Routines to construct queries, send them to the server, and process the results.
Status- and error-reporting functions for determining the exact reason for an error when other C API calls fail.

Creating MySQL database on Windows system:

- Start the MySQL server at the prompt in c:\mysql\bin. Refer the previous session Installing MySQL on Windows for further details.
- Now invoke the mysql client program by typing mysql at the prompt.
- The prompt is changed to a mysql> prompt.

CREATE DATABASE :

CREATE DATABASE creates a database with the given name. To use this statement, you need the CREATE privilege for the database. CREATE SCHEMA is a synonym for CREATE DATABASE,
Like this CREATE {DATABASE | SCHEMA}.
CREATE TABLE

CREATE TABLE creates a table with the given name. You must have the CREATE privilege for the table

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   (create_definition,...)
   [table_option] ...
   [partition_options]
```

Like this:
```
CREATE TABLE example (
id      INT UNSIGNED NOT NULL,
name    VARCHAR(100) NOT NULL,
type    VARCHAR(20) NOT NULL,
size    INT UNSIGNED NOT NULL,
description  VARCHAR(200),
author  VARCHAR(50),
data    MEDIUMBLOB NOT NULL,
PRIMARY KEY (id)
        );
```

6 The MySQL C API Application

Here we show a simple C++ program which makes a lot of functions, So first we look how we could make Mysql works to let him communicate with C++ application and max allow 32MB and let him use the ssl certificate and keys, sure we could use more than 32 MB, but for our files till now we don't need more than 32 MB.
So we used this command to make this function

mysqld --max_allowed_packet=32MB --ssl-ca=cacert.pem --ssl-cert=server_crt.pem - -ssl-key=server_key.pem

and we can see it on Figure 6.1 Mysql Setup.

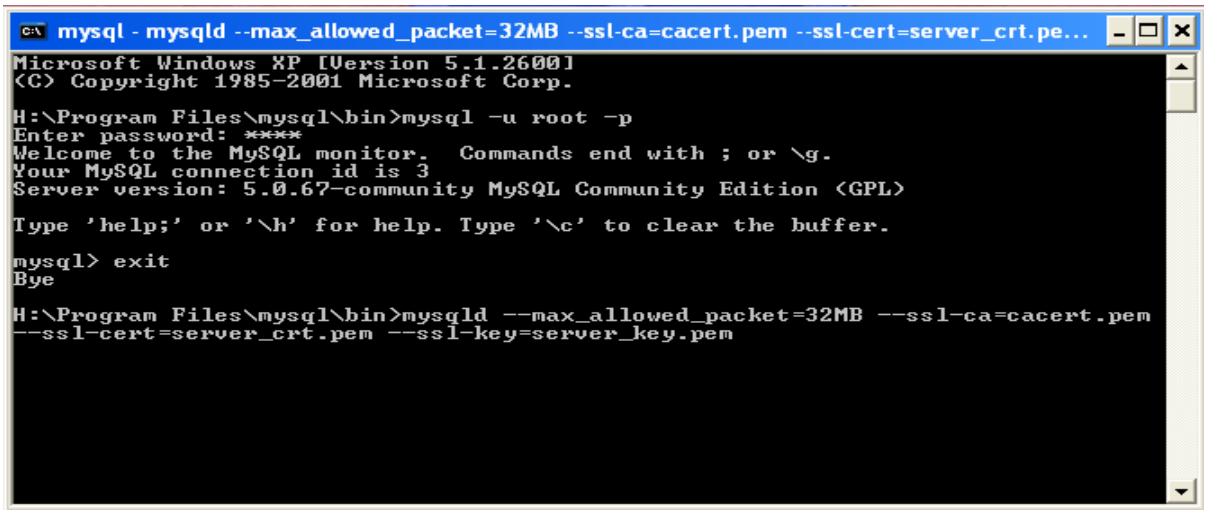


Figure 6.1 Mysql Setup.

We could also make a tables in database by mysql commands but for easy usage we preferred to use it directly in C++ application

```
CREATE TABLE test       (
            id      INT UNSIGNED NOT NULL,
            name    VARCHAR(100) NOT NULL,
            type    VARCHAR(20) NOT NULL,
            size    INT UNSIGNED NOT NULL,
            description  VARCHAR(200),
            author  VARCHAR(50),
            data    MEDIUMBLOB NOT NULL,
            PRIMARY KEY (id)
                        );

CREATE TABLE servers   (
            address       VARCHAR(50),
            name          VARCHAR(100),
            description   VARCHAR(200),
            PRIMARY KEY (address, name)
                        );
```

And on Figure 6.2 We have a basic C++ application and how it looks like, the important thing that we could make a server like a localhost so we can communicate with our database which runs on our computer, or we can make it like ip address if we want it to communicate with distant Server, and also we must enter a username and a password to let the program communicate with its database.

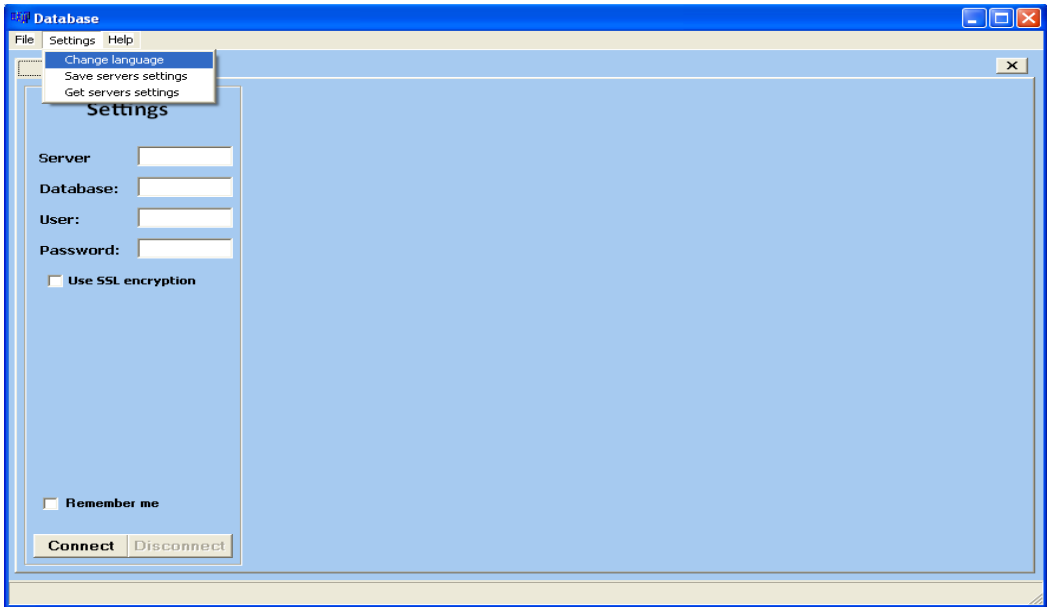


Figure 6.2 C++ application.

So on Figure 6.3 we have a C++ application without ssl connection and we can make a lot of functions with it like download and erase files and change language between English and Czech and also giving a description of files which we download and also a name and description for a database for specific purpose.

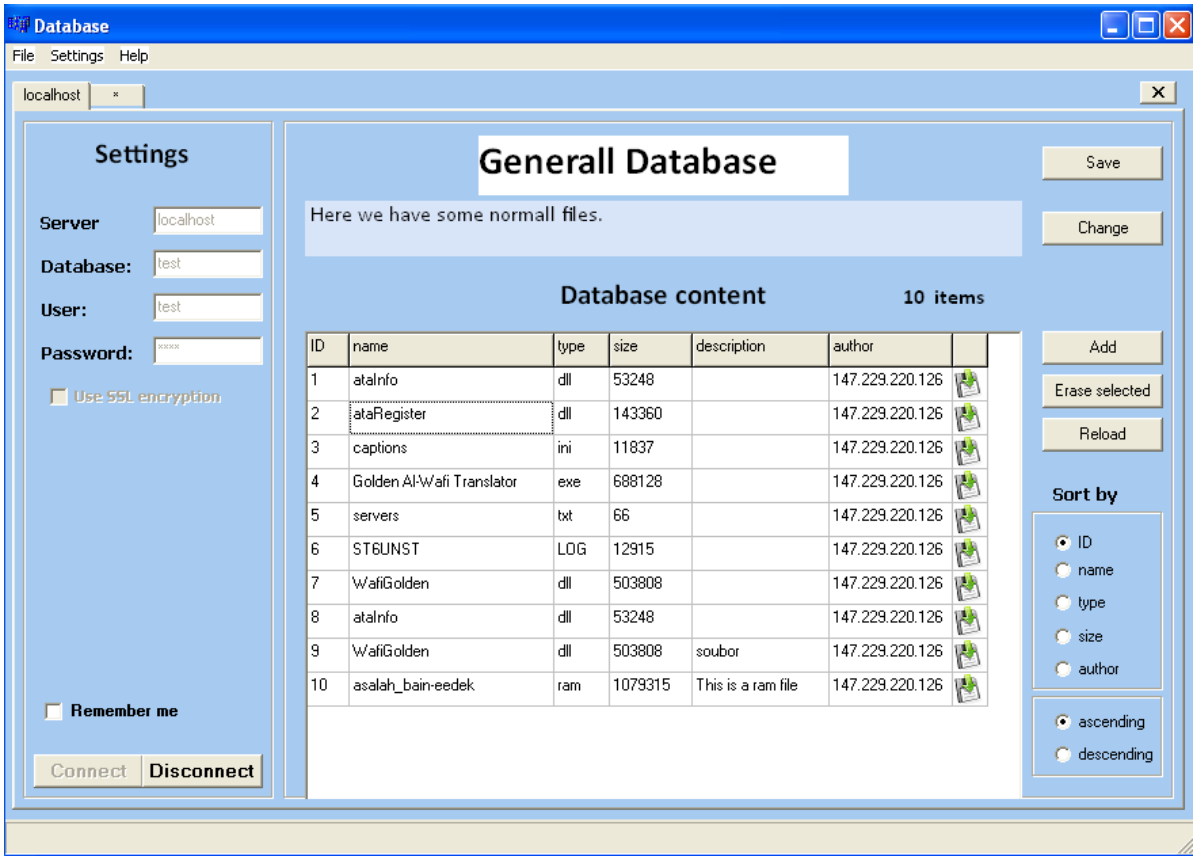


Figure 6.3 C++ application without SSL.

As we see on Figure 6.4 a C++ application with ssl encryption we must specify the location of the certificate to make a program use a ssl connection, and also we could see the type of the files which specify the extension and the size of it and also its description and a ip address from which ip we make downloading.

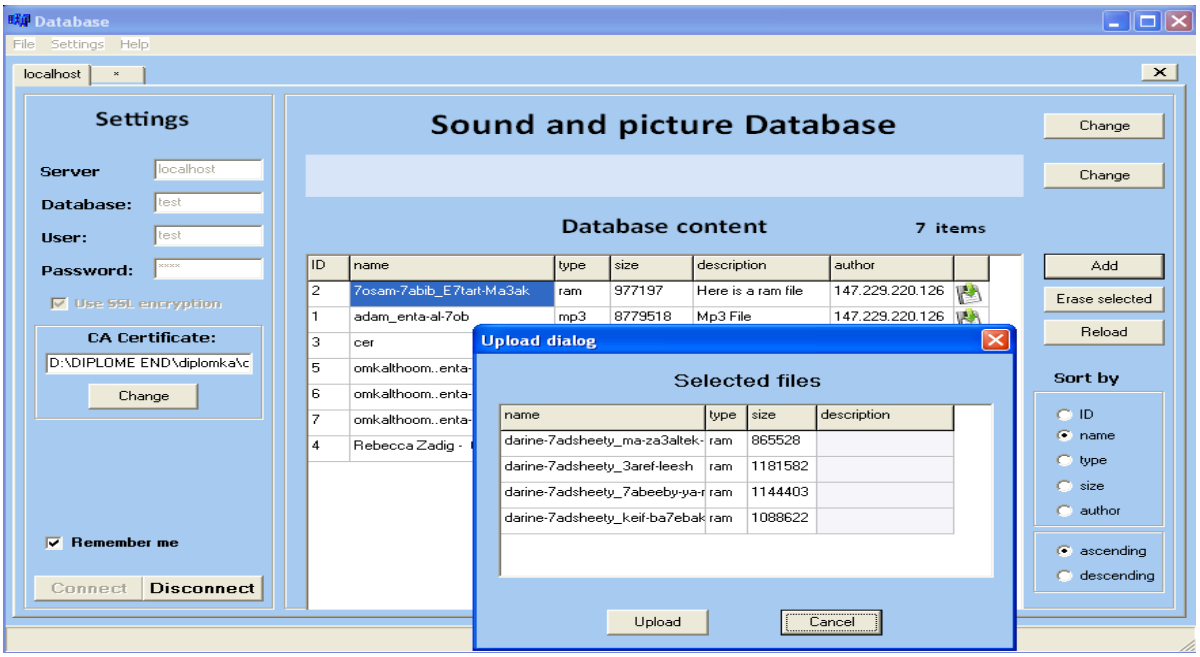


Figure 6.4 C++ application with SSL

7 What is SoX

SoX reads and writes audio files in most popular formats and can optionally apply effects to them; it can combine multiple input sources, synthesise audio, and, on many systems, act as a general purpose audio player or a multi-track audio recorder. It also has limited ability to split the input in to multiple output files.
Almost all SoX functionality is available using just the sox command, however, to simplify playing and recording audio, if SoX is invoked as play the output file is automatically set to be the default sound device and if invoked as rec the default sound device is used as an input source. Additionally, the soxi command provides a convenient way to just query audio file header information.
SoX is a command-line audio processing tool, particularly suited to making quick, simple edits and to batch processing.

SoX processing chain can be summarised as follows:

Input(s) >> Combiner >> Effects >> Output(s)

For example:
If we want to change from au format to wav format so we do this command

sox file.au file.wav

libsox is a library of sound sample file format readers/writers and sound effects processors. It is mainly developed for use by SoX but is useful for any sound application.

sox_format_init function performs some required initialization related to all file format handlers. If compiled with dynamic library support then this will detect and initialize all external libraries. This should be called before any other file operations are performed.

sox_format_quit function performs some required cleanup related to all file format handlers.
sox_open_input function opens the file for reading whose name is the string pointed to by path and associates an sox_format_t with it. If info is non-NULL then it will be used to specify the data format of the input file. This is normally only needed for headerless audio files since the information is not stored in the file. If filetype is non-NULL then it will be used to specify the file type. If this is not specified then the file type is attempted to be derived by looking at the file header and/or the filename extension. A special name of "-" can be used to read data from stdin.

sox_open_output function opens the file for writing whose name is the string pointed to by path and associates an sox_format_t with it. If info is non-NULL then it will be used to specify the data format of the output file. Since most file formats can write data in different data formats, this generally has to be specified. The info structure from the input format handler can be specified to copy data over in the same format. If comment is non-NULL, it will be written in the file header for formats that support comments. If filetype is non-NULL then it will be used to specify the file type. If this is not specified then the file type is attempted to be derived by looking at the filename extension. A special name of "-" can be used to write data to stdout.

The function sox_read reads len samples in to buf using the format handler specified by ft. All data read is converted to 32-bit signed samples before being placed in to buf. The value of len is specified in total samples. If its value is not evenly divisable by the number of channels, undefined behavior will occur.
The function sox_write writes len samples from buf using the format handler specified by ft. Data in buf must be 32-bit signed samples and will be converted during the write process. The value of len is specified in total samples. If its value is not evenly divisable by the number of channels, undefined behavior will occur.
The sox_close function dissociates the named sox_format_t from its underlying file or set of functions. If the format handler was being used for output, any buffered data is written first.
The function sox_find_effect finds effect name, returning a pointer to its sox_effect_handler_t if it exists, and NULL otherwise.
The function sox_create_effect instantiates an effect into a sox_effect_t given a sox_effect_handler_t *. Any missing methods are automatically set to the corresponding nothing method.
The function sox_effect_options allows passing options into the effect to control its behavior. It will return SOX_EOF if there were any invalid options passed in. On success, the effp->in_signal will optional contain the rate and channel count it requires input data from and effp->out_signal will optionally contain the rate and channel count it outputs in. When present, this information should be used to make sure appropriate effects are placed in the effects chain to handle any needed conversions.
Passing in options is currently only supported when they are passed in before the effect is ever started. The behavior is undefined if its called once the effect is started.
sox_create_effects_chain will instantiate an effects chain that effects can be added to. in_enc and out_enc are the signal encoding of the input and output of the chain respectively. The pointers to in_enc and out_enc are stored internally and so their memory should not be freed. Also, it is OK if their values change over time to reflect new input or output encodings as they are referenced only as effects start up or are restarted.
sox_delete_effects_chain will release any resources reserved during the creation of the chain. This will also call sox_delete_effects if any effects are still in the chain.
sox_add_effect adds an effect to the chain. in specifies the input signal info for this effect. out is a suggestion as to what the output signal should be but depending on the effects given options and on in the effect can choose to do differently. Whatever output rate and channels the effect does produce are written back to in. It is meant that in be stored and passed to each new call to sox_add_effect so that changes will be propagated to each new effect.

There are a number of advantages that open source code offers over closed source:

- Bug fixing

Almost all software releases contain bugs. Hopefully, the people developing the software will have spotted and dealt with anything obvious, but any development team has only so much time in which to test a piece of software before it is released.

When a bug is spotted in proprietary software, the only people who can fix it are the original developers, as only they have access to the source code. Open source software is different. As a large number of users can access and change the code, bugs tend to

be more visible and more rapidly corrected. One of the slogans of the open source movement is that ‘given enough eyeballs, all bugs are shallow’ [Eric Raymond, The Cathedral and the Bazaar].

- Security

  Having access to the source code allows the user of that software to choose the approach to security that they want. In other words it allows you to take ownership of your own security. It also enables certains approaches that are not available with closed source and it is possible to decide on your own security priorities and to allocate resources accordingly.

  Access to source code makes it easier to detect security flaws in software, whether you are looking to fix them or exploit them. Given that most users do not tend to seek out security gaps until they have a pressing reason to do so, this would seem to suggest that open source code is less secure than closed source.

  In practice, however, the skills and time required to find security flaws, work out how they can be exploited, and then initiate an attack, are more specialized than the mundane debugging skills required to close exploits. Given the number of programmers who can access and edit open source code, compared with the few that are entitled to access closed source code, it should not come as a surprise that flaws in open source software tend to be fixed more rapidly, before serious damage can be done. Fixing holes in closed source software usually depends on the problem being of a high enough priority to command the immediate attention of the original development team.

- Customization

  Closed source applications can only be customized or adapted within the scope provided by the original vendor but never outside its boundaries. Open source applications may be customized by anyone with the requisite skill. Thus, open source software can be readily adapted to meet specific user needs. Even if you cannot program yourself, if you would like something added or customized you can generally pay an appropriately skilled software developer to do it for you.

  For businesses or educational institutions, the ability to customize source code may enable improvements to the ‘best practice’ provided by default installations, therefore improving efficiency and possibly providing a competitive advantage.

  The Open University made a decision in 2005 to invest a substantial sum of money in developing the Moodle virtual learning environment to best suit their requirements. As the Moodle source code is open, they can do this for themselves rather than having to persuade a commercial vendor to do so on their behalf.

- Translation

  With access to the source code it is easy to translate the language of the software interface. Large closed source commercial software vendors are usually unwilling to translate their products into less widely spoken languages, as the market for them would be too small to guarantee profit.

  An example of this is the regional government of the South Tyrol, who developed a version of OpenOffice in the local Ladin language, which has around 30,000 speakers. This is too small a number to be worth commercial investment, but culturally important in terms of the survival of the language.

- Avoiding lock-in

  Organisations are said to be ‘locked-in’ to software products when the costs of switching to alternatives are prohibitively high.

  Proprietary software vendors can ‘lock’ users in to their products by ensuring that they are not readily compatible with potential rivals. Vendors may then increase the price of product upgrades or support without too great a risk of losing existing customers.

  As there is no incentive to use non-standard formats to inhibit compatibility, open source software tends to use open standard formats and there is little danger of being ‘locked-in’ by a vendor. Even when non-standard formats are used in open-source code, it is always possible to document them from the source code. On the contrary, closed formats used by proprietary software need to be reverse-enginered, a burdensome and expensive process that may need to be repeated when the format is subsequently changed.

  It should be admitted, of course, that open source software does not come without switching costs of its own. Some administrative and re-training costs must be borne by any organisation that opts to switch between different software. And proprietary software may use open standards too, as is the case with Adobe's Acrobat Reader, a closed-source programme to read PDF files (PDF format is an open standard). Indeed the use of open standards is especially important in ensuring future access to data as it will be possible to find or create alternative programs that conform to the standard.

- Mitigation of vendor collapse or product discontinuation

  Commercial software vendors go bust or get bought up from time to time. When this happens there is no guarantee that their software products will continue to be available, supported, or updated. This can result in users needing to switch products, which can be very expensive and difficult, especially if they were heavily 'locked-in' to their current product.

  Even with healthy companies, new releases often mean that older software and format versions are discontinued and no longer supported.

  With open source software, this danger is greatly reduced. As the source code is not 'owned' in the same way that proprietary source code is, it may be picked up and developed by anyone with an interest in a product's survival. Unless you are part of an organization with very significant technical resources, you are unlikely to want to take on full responsibility for this, but, thanks to the way in which successful open source projects gather user communities about them, there will probably be other interested parties to share these responsibilities.

- Learning from examples

  If you are interested in programming, open source code provides an excellent resource from which to learn, and open source projects provide a practical environment in which to test your skills. Just watching the development process can provide an education in itself. If you choose to submit code to an open source project, it will generally be checked and commented on by experienced programmers. Once you have convinced the project community that your code is of appropriate quality, you may be granted full committer rights yourself.

  Ross Gardler of OSS Watch, and a member of The Apache Software Foundation, claims to ‘have learnt far more through open source than through any form of formal education or [software development] contract work.

- Being part of a community

  By adopting open source software you become part of a community of users and developers who have an interest in working together to support each other and improve the software. The extent to which you engage with this community is up to you, but you may obtain the intangible benefits of goodwill if you do.

  Programmers in particular can benefit from belonging to an open source community. It can help establish reputation and respect, as well as gaining valuable experience.

- Cost

  Many open source programs can be obtained at no cost or with a very low cost. This is often an important issue for individuals and in many cases this has been the main reason for an individual adopting a particular open source solution over a closed source alternative.

  However, other costs may arise: training, consulting, maintenance, etc. As a result the total cost of ownership may not differ between a closed source solution and an open source alternative for institutions. However, in some particular markets the difference in price can be significant between a closed source solution and an open source solution.

7.1 Soxi

Soxi is for displaying informatik from the header of a given audio file or files.

OPTIONS:

- -V Set verbosity.
- -t Show detected file-type.
- -r Show sample-rate.
- -c Show number of channels.
- -s Show number of samples (0 if unavailable).
- -d Show duration in hours, minutes and seconds (0 if unavailable). Equivalent to number of samples divided by the sample-rate.
- -b Show number of bits per sample.
- -e Show the name of the audio encoding.
- -a Show file comments (annotations) if available

7.2 Format Types in sox

In sox there are two types of audio file format that can work with. The first is self-describing; these formats include a header that completely describes the characteristics of the

audio data that follows. The second type is `headerless' (or raw data); here, the audio data characteristics must be described using the SoX command line.

Here we describe the four characteristics that describe the format of audio data such that it can be processed with SoX :

- sample rate

  The sample rate in samples per second (`Hertz' or `Hz'). For example, digital telephony traditionally uses a sample rate of 8000 Hz (8 kHz); audio Compact Discs use 44100 Hz (44.1 kHz); Digital Audio Tape and many computer systems use 48 kHz; professional audio systems typically use 96 or 192 kHz.

- sample size

  The number of bits used to store each sample. The most popular is 16-bit (two bytes); 8-bit (one byte) was popular in the early days of computer audio, and is still used in telephony; 24-bit (three bytes) is used, primarily as an intermediate format, in the professional audio arena. Other sizes are also used.

- data encoding

  The way in which each audio sample is represented (or `encoded'). Some encodings have variants with different byte-orderings or bit-orderings; some `compress' the audio data, i.e. the stored audio data takes up less space (i.e. disk-space or transmission band-width) than the other format parameters and the number of samples would imply. Commonly-used encoding types include floating-point, μ-law, ADPCM, signed-integer PCM, and FLAC.

- channels

  The number of audio channels contained in the file. One (`mono') and two (`stereo') are widely used. `Surround sound' audio typically contains six or more channels.

8 Audio files

Audio files are sound files, or files that play a sound when clicked on. One of the most common audio formats is the wave file, or [filenamehere].wav. Another is the MP3 file, or [filenamehere].mp3. The type of audio format used is indicated by the file extension – the last three letters following the dot. Standard audio players included with operating systems will play common types of audio files, but more exotic sound formats might require downloading codecs for the player to extend its capabilities.

Types of formats:

It is important to distinguish between a file format and a codec. A codec performs the encoding and decoding of the raw audio data while the data itself is stored in a file with a specific audio file format. Though most audio file formats support only one audio codec, a file format may support multiple codecs, as AVI does.
There are three major groups of audio file formats:

- Uncompressed audio formats, such as WAV, AIFF and AU;
- formats with lossless compression, such as FLAC, Monkey's Audio (filename extension APE), WavPack (filename extension WV), Shorten, Tom's lossless Audio Kompressor (TAK), TTA, ATRAC Advanced Lossless, Apple Lossless and lossless Windows Media Audio (WMA).
- formats with lossy compression, such as MP3, Vorbis, Musepack, ATRAC, lossy Windows Media Audio (WMA) and AAC

8.1 AIFF (Audio Interchange File Format)

File format used by Macintosh computers and Silicon Graphics Incorporated to store and transmit high-quality audio data such as music. It can store monaural, stereo or multi-channel sound as used for soundtracks. Apple Computer developed the AIFF format in 1987-88 in accordance with Electronic Arts Interchange File Format (IFF) standards. For this reason, AIFF files might have the extension .aif or .ief.
AIFF files are uncompressed, making the files quite large compared to the ubiquitous MP3 format. AIFF files are comparable to Microsoft's wave files, because they are high quality they are excellent for burning to CD.
The structure of an AIFF file contains distinct blocks of data called "chunks." Each chunk contains specific information about the file's contents.

8.1 The Au file format

Simple audio file format introduced by Sun Microsystems. The format was common on NeXT systems and on early web pages. Originally it was headerless, being simply 8-bit µ-law-encoded data at an 8000 Hz sample rate. Hardware from other vendors often used sample rates as high as 8192 Hz, often integer factors of video clock signals. Newer files have a header that consists of six 32-bit words, an optional information chunk and then the data (in big endian format).
Although the format now supports many audio encoding formats, it remains associated with the µ-law logarithmic encoding

8.2 CD-DA

An abbreviation for Compact Disc Digital Audio. It is also known as “Red Book Audio” and is the digital sound format used by audio CDs. Data is encoded by starting with a source sound file, and sampling it to convert it to digital format. CD-DA audio uses a sample rate of 44.1 kHz, which is roughly double the highest frequency audible by humans. Each sample is 16 bits in size, and the sampling is done in stereo. Therefore, each second of sound takes (44,100 * 2 * 2) bytes of data, which is 176,400 bytes.

8.3 A waveform audio file

Also known as a wave file, or simply .wav after its extension, is a common type of sound file. Microsoft and IBM introduced the wav file in 1991 for use on the Microsoft Windows 3.1 operation system (OS). Long before digital audio became a staple, computer users were exposed to the wav file as an embedded sound file that played a chime-like sound at boot up of the Windows operating system.
The wav file had two very big things going for it when introduced. Firstly, it could digitize sounds 100% faithful to the original source because it is a lossless format. “Lossless” means that the wav file format does not compromise audio quality even when it holds compressed data. Secondly, the wav file is very easy to edit and manipulate with software .

8.4 MP3

Digital audio file compressed with a standard defined by the Motion Pictures Experts Group (MPEG). MPEG was formed to develop techniques for dealing with digital video; since most video also contains audio, MP3 was developed as an audio extension of that work. Officially known as "MPEG-1, Layer 3", MP3 is a lossy compression algorithm that uses psychoacoustic modeling to reduce the size of audio files by up to 90%.
Psychoacoustics takes advantage of deficiencies in the human hearing system to throw away digital bits corresponding to sounds that cannot be heard. The human ear cannot hear soft sounds in the presence of loud sounds having a similar frequency; for example, a voice conversation becomes inaudible when a jet flies low overhead. This effect is known as auditory masking, and done correctly the discarded sounds will not be missed.
MP3 is a lossy algorithm in the sense that the original bits cannot be recreated from the compressed bits. In terms of hearing, however, MP3 is lossless because the human ear cannot distinguish between a CD recording and a properly encoded MP3 version of it. MP3s achieve this transparency at a bit rate of approximately 256 kilobits per second, or roughly one sixth of the 1.4 megabits per second required by the compact disc format.
MP3s can be recorded at lower bit rates, saving even more space, but audible differences begin to appear at rates below 128 kilobits per second. At these lower bit rates, MP3 can use a trick known as joint stereo to improve quality. Audio generally consists of left and right audio tracks. Joint stereo combines, whenever possible, the sounds common to both left and right tracks into one track. Instead of left and right, it has "common" and "different" channels.

8.5 Windows Media Audio

Digital music format that was developed by Microsoft. It is an audio compression format that has 4 codecs: WMA, WMA Pro, WMA Lossless, and WMA Voice. Usually, WMA files come in ASF (Advanced Systems Format) containers which have the ability to be encrypted with DRM – the WMA system on its own doesn’t support DRM. Once in this container, metadata (equivalent to the MP3 ID3 tag) can also be included: song name, artist, track number etc.

8.6 TTA

Stands for The True Audio :It's a free realtime lossless audio compressor based on adaptive prognostic filters. TTA offers adequate compression levels while maintaining high operation speeds.

Having no loss in data or quality, TTA compresses files to as little as 30% of their original size (allows for the storage of up to 20 audio CDs worth of music on a single DVD-R). Minimal system requirements has made it quite popular.Compression ratios achieved by the TTA codec vary, depending on music type, but range from 30% - 70% of the original. The TTA lossless compressed audio format supports both ID3v1 and ID3v2 information tags. The compression is performed on multichannel 8, 16 and 24-bit data of WAV audio files.

8.7 AAC

Stands for advanced audio coding and is the logical successor to MP3 (ISO/MPEG Audio Layer-3) for audio coding at medium to high bit rates.
The idea of AAC's algorythm is to exploit two primary coding strategies to dramatically reduce the amount of data needed to convey high-quality digital audio. AAC has the sampling frequencies between 8 Hz and 96 kHz and any number of channels between 1 and 48. AAC uses the modified discrete cosine transform (MDCT) together with the increased window lengths of 2,048 points. AAC is much more capable of encoding audio with streams of complex pulses and square waves than MP3 or Musicam. AAC can be switched dynamically between MDCT block lengths of 2,048 points to 256 points.
AAC takes a modular approach to encoding. Depending on the complexity of the bitstream to be encoded, the desired performance and the acceptable output, implementers may create profiles to define which of a specific set of tools they want use for a particular application. So the AAC LD coding scheme bridges the gap between speech coding schemes and high quality audio coding schemes.

8.8 Real Audio

Real audio is an audio format. It was first introduced by RealNetworks in 1995, the current version being Real Audio 10. Files of this format usually have the extensions of RA, RAX, RM or RAM.
Real Audio is often used as a streaming audio format (can be played at the same time as it's downloaded) due to its ability to conform to low bandwidths. This makes Real Audio a common format for many internet radio stations.

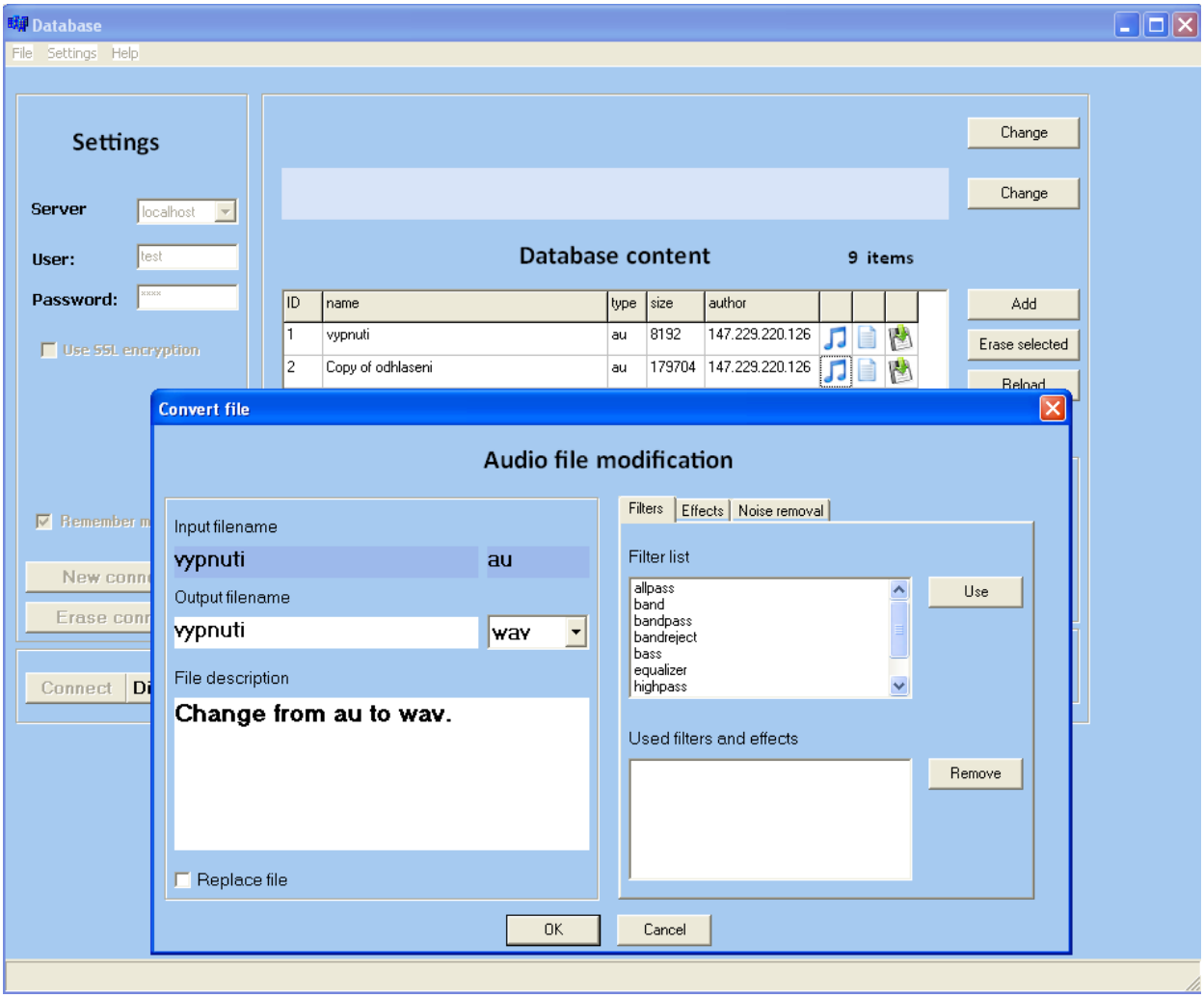


Figure 8.1 Audio formats.

In the figure 8.1, Sox program change the input file to other types of format , which could be download it late and use them for other purposes.
Our example show how to change from au format to wav format.
Sox program support alot of types of format but unfortunately in Windows doesnt support all,and makes some errors when using them, for example Mp3 format couldnot work well in our Program, but it supports it in unix platform.

9 Audio effect

Sound effects or audio effects are artificially created or enhanced sounds, or sound processes used to emphasize artistic or other content of films, television shows, live performance, animation, video games, music, or other media.
In motion picture and television production, a sound effect is a sound recorded and presented to make a specific storytelling or creative point without the use of dialogue or music. The term often refers to a process applied to a recording, without necessarily referring to the recording itself.
In professional motion picture and television production, dialogue, music, and sound effects recordings are treated as separate elements. Dialogue and music recordings are never referred to as sound effects, even though the processes applied to them, such as reverberation or flanging effects, often are called "sound effects".

9.1 A graphic equalizer

Is a high-fidelity audio control that allows the user to see graphically and control individually a number of different frequency bands in a stereophonic system. A typical graphic equalizer consists of several audio filter/amplifiers, each centered at a specific frequency in the audio range. Most graphic equalizers have two identical sets of filter/amplifiers, one for each channel in a stereophonic system.

9.2 The gain (volume)

It controls in most graphic equalizers are slide potentiometers that are adjusted by moving a control button up or down. Gain is increased by sliding the button upwards. The slide potentiometers for each channel are placed side-by-side, with the lowest-frequency unit at the left and the highest-frequency unit at the right. In this way, the positions of the buttons appear to follow a graphical curve that represents the gain as a function of frequency for each channel.

9.3 Graphic equalizers

Equalizer are common in middle- and high-end stereophonic sound systems for consumer use. They are found in practically all professional recording studios. Graphic equalizer programs are also available for use in fine-tuning sound in a personal computer.

9.4 Chorus

A chorus is a Delay for which the delay time is modulated by an LFO (Low Frequency Oscillator). The most used LFO waveforms are sine and triangle.

Most modern chorus effect unit, be they plugins or hardware based, also spread the effect signal across the stereo field. Sometimes a parameter to control the depth and other behaviours may be encountered.

The effect can produce interesting sounds, one of which is to produce the illusion of more than one instance of the instrument/vocalist being present. What you get out of this effect, largely depends on your desire to experiment.

The effect is a type of delay , the original signal is duplicated and played at varying lengths and pitches. This creates the effect of multiple sources, as each source is very slightly out of time and tune (just as in real life). Technically, a chorus is similar to a flanger.

The chorus effect can be simulated by signal processing equipment. The signal processor may be software running on a computer, a ROM-encoded effect in a digital effect processor, or an analog effect processor. If the processor is hardware-based, it may be packaged as a foot pedal, a rack-mount module, a table-top device, or built in to an instrument amplifier.

Regardless of the technology or form factor, the processor achieves the effect by taking an audio signal and mixing it with one or more delayed, pitch-modulated copies of itself. The pitch of the added voices is typically modulated by an LFO, which makes the overall effect similar to that of a flanger, except with longer delays and without feedback.

Stereo chorus effect processors produce the same effect, but it is varied between the left and right channels by offsetting the delay or phase of the LFO. The effect is thereby enhanced because sounds are produced from multiple locations in the stereo field. Used on instruments like "clean" (undistorted) electric guitar and keyboards, it can yield very dreamy or ambient sounds. Commercial chorus effect devices often include controls that enable them to be used to also produce delay, reverberation, or other related effects that use similar hardware, rather than exclusively as chorus effects.

9.5 Flange

Flanger is related to the phasing effect produced by a, well, phaser effects unit. It is produced when two identical signals are mixed together, with one of the signals time-delayed by a small and gradually changing amount. The amount is usually equal to or less than 20 milliseconds. Peaks and notches are produced in the combined frequency spectrum, related in a linear harmonic series. Part of the output signal is fed back in and resonates, intensifying the peaks and notches. This effect was originally generated with 3 three headed tape machines. Two of the tape machines would play the signal, obviously somewhat out of synch, and the third tape machine would record the output. The modern version of the effect is created using DSP (digital signal processing) technology.

Flange is an audio effect caused by mixing the original audio with a copy that is delayed by a varying amount that cycles over time. The frequency of the copy is also offset by an amount related to the delay. Chorus uses a larger delay, to make one voice or instrument sound like many.

10 Effects in Sox

SoX can be used to invoke a number of audio `effects'. Multiple effects may be applied by specifying them one after another at the end of the SoX command line; forming an effects chain. Note that applying multiple effects in real-time (i.e. when playing audio) is likely to need a high performance computer; stopping other applications may alleviate performance issues should they occur.

- allpass frequency[k] width[h|k|o|q]

Apply a two-pole all-pass filter with central frequency (in Hz) frequency, and filter-width width. An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship.
Where h is hertz and k is khz and o is ocatves and q is q factor

- band [-n] center[k] [width[h|k|o|q]]

Apply a band-pass filter. The frequency response drops logarithmically around the center frequency. The width parameter gives the slope of the drop. The frequencies at center + width and center - width will be half of their original amplitudes. band defaults to a mode oriented to pitched audio, i.e. voice, singing, or instrumental music. The -n (for noise) option uses the alternate mode for un-pitched audio.

- bandpass|bandreject [-c] frequency[k] width[h|k|o|q]

Apply a two-pole Butterworth band-pass or band-reject filter with central frequency frequency, and (3dB-point) band-width width. The -c option applies only to bandpass and selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain.

- bandreject frequency[k] width[h|k|o|q]

Apply a band-reject filter. See the description of the bandpass effect for details.

- bass|treble gain [frequency[k] [width[s|h|k|o|q]]]

Boost or cut the bass (lower) or treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

- delay {length}

Delay one or more audio channels. length can specify a time or, if appended with an ‘s’, a number of samples. Do not specify both time and samples delays in the same command. For example, delay 1.5 0 0.5 delays the first channel by 1.5 seconds, the third channel by 0.5 seconds, and leaves the second channel (and any other channels that may be present) un-

delayed. As showed in the Figure 10.1 below,which we can change the parametrs of the delay in channels.

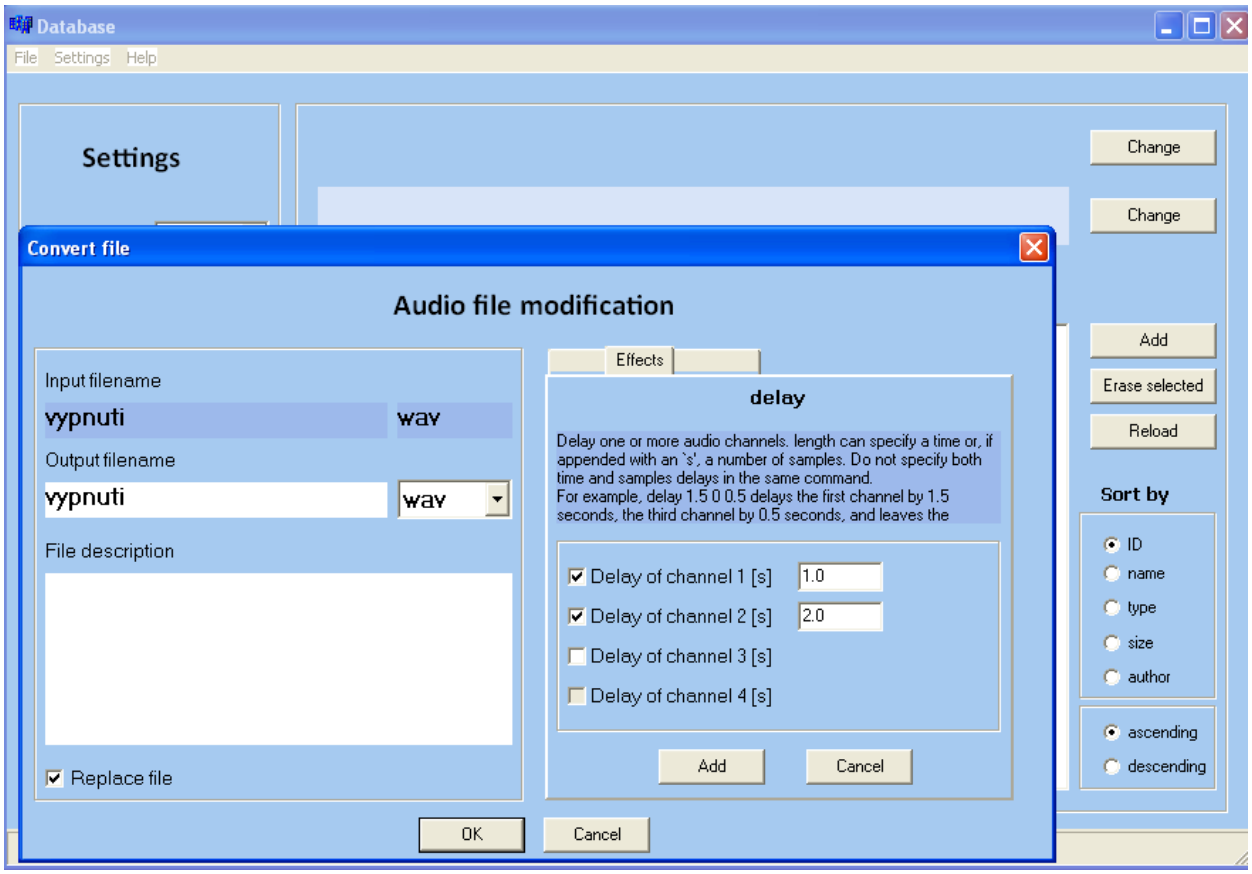


Figure 10.1 Delay Effect

- dither [−r|−t] [−s|−f filter] [depth]

Apply dithering to the audio. Dithering deliberately adds a small amount of noise to the signal in order to mask audible quantization effects that can occur if the output sample size is less than 24 bits. The default (or with the −t option) is Triangular (TPDF) white noise. The −r option can be used to select Rectangular Probability Density Function (RPDF) white noise. Noise-shaping (only for certain sample rates) can be selected with −s. With the −f option, it is possible to select a particular noise-shaping filter from the following list: lipshitz, f-weighted, modified-e-weighted,improved-e-weighted, gesemann, shibata, low-shibata, high-shibata. Note that most filter types are available only with 44100Hz sample rate. The filter types are distiguished by the following
properties: audibility of noise, level of (inaudible, but in some circumstances, otherwise problematic) shaped high frequency noise, and processing speed.

- earwax

Makes audio easier to listen to on headphones. Adds ‘cues’ to 44.1kHz stereo (i.e. audio CD format)
audio so that when listened to on headphones the stereo image is moved from inside your

head (standard for headphones) to outside and in front of the listener (standard for speakers).

- echo gain-in gain-out <delay decay>

Add echoing to the audio. Echoes are reflected sound and can occur naturally amongst mountains (and sometimes large buildings) when talking or shouting; digital echo effects emulate this behaviour and are often used to help fill out the sound of a single instrument or vocal. The time difference between the original signal and the reflection is the 'delay' (time), and the loudness of the relected signal is the 'decay'. Multiple echoes can have different delays and decays.
Each given delay decay pair gives the delay in milliseconds and the decay (relative to gain-in) of that echo. Gain-out is the volume of the output.

- equalizer frequency[k] width[q|o|h|k] gain

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike band-pass and band-reject filters) that at all other frequencies is unchanged.
frequency gives the filter's central frequency in Hz, width, the band-width, and gain the required gain or attenuation in dB.

- fade [type] fade-in-length [stop-time [fade-out-length]]

Add a fade effect to the beginning, end, or both of the audio.
For fade-ins, this starts from the first sample and ramps the volume of the audio from 0 to full volume over fade-in-length seconds. Specify 0 seconds if no fade-in is wanted.
For fade-outs, the audio will be truncated at stop-time and the volume will be ramped from full volume down to 0 starting at fade-out-length seconds before the stop-time. If fade-out-length is not specified, it defaults to the same value as fade-in-length. No fade-out is performed if stop-time is not specified. If the file length can be determined from the input file header and length-changing effects are not in effect, then 0 may be specified for stop-time to indicate the usual case of a fade-out that ends at the end of the input audio stream.
All times can be specified in either periods of time or sample counts. To specify time periods use
the format hh:mm:ss.frac format. To specify using sample counts, specify the number of samples and append the letter 's' to the sample count (for example '8000s').
An optional type can be specified to change the type of envelope. Choices are q for quarter of a sine wave, h for half a sine wave, t for linear slope, l for logarithmic, and p for inverted parabola. The default is logarithmic.

- repeat count

Repeat the entire audio count times. Requires temporary file space to store the audio to be repeated. Note that repeating once yields two copies: the original audio and the repeated audio.

- reverse

Reverse the audio completely. Requires temporary file space to store the audio to be reversed.

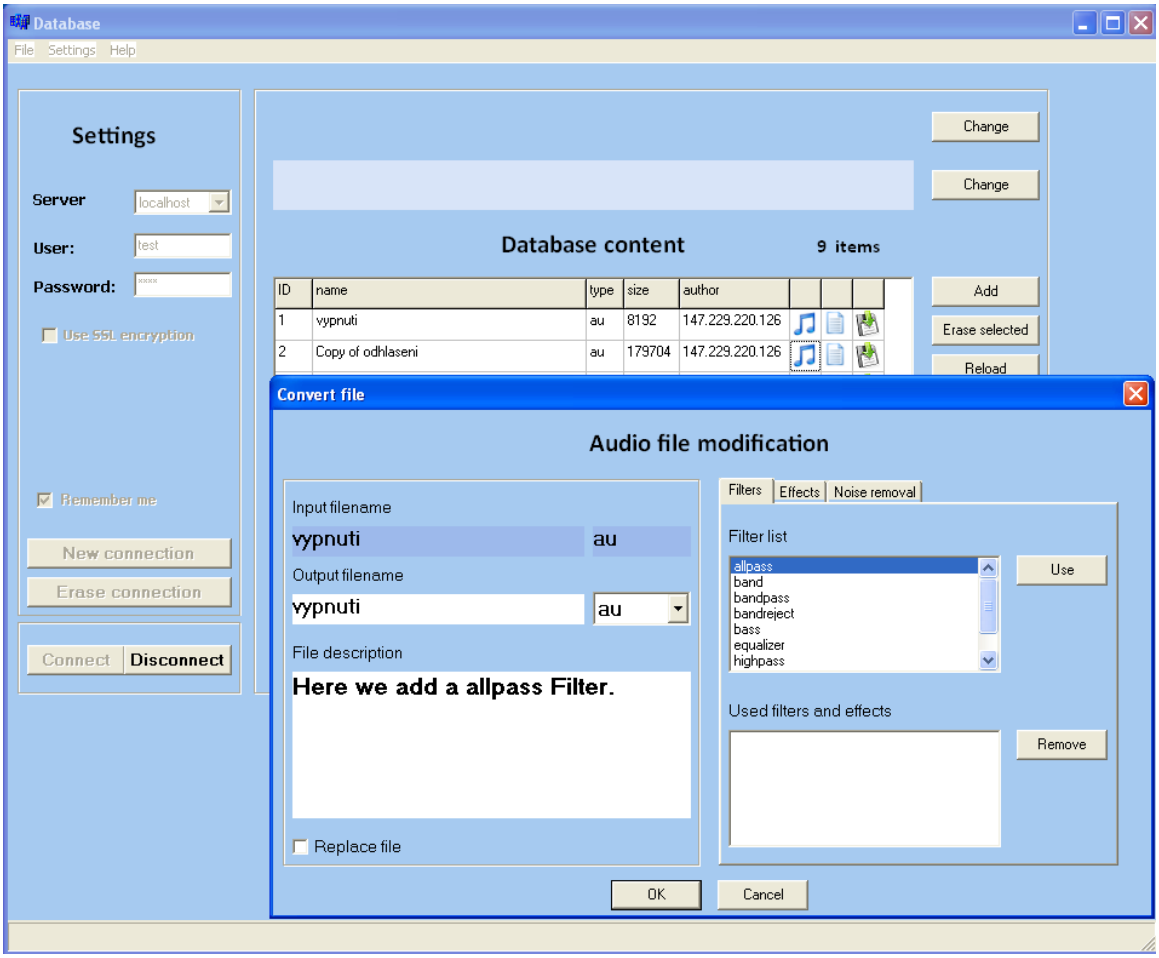


Figure 10.2 Filter

We can make in our program some types of filtres which sox provide, for example allpass,highpass,lowpass and so on.

Filters Have some parametrs that should be included to give the best resultes on sound files ,for example a frequancy and bandwidth

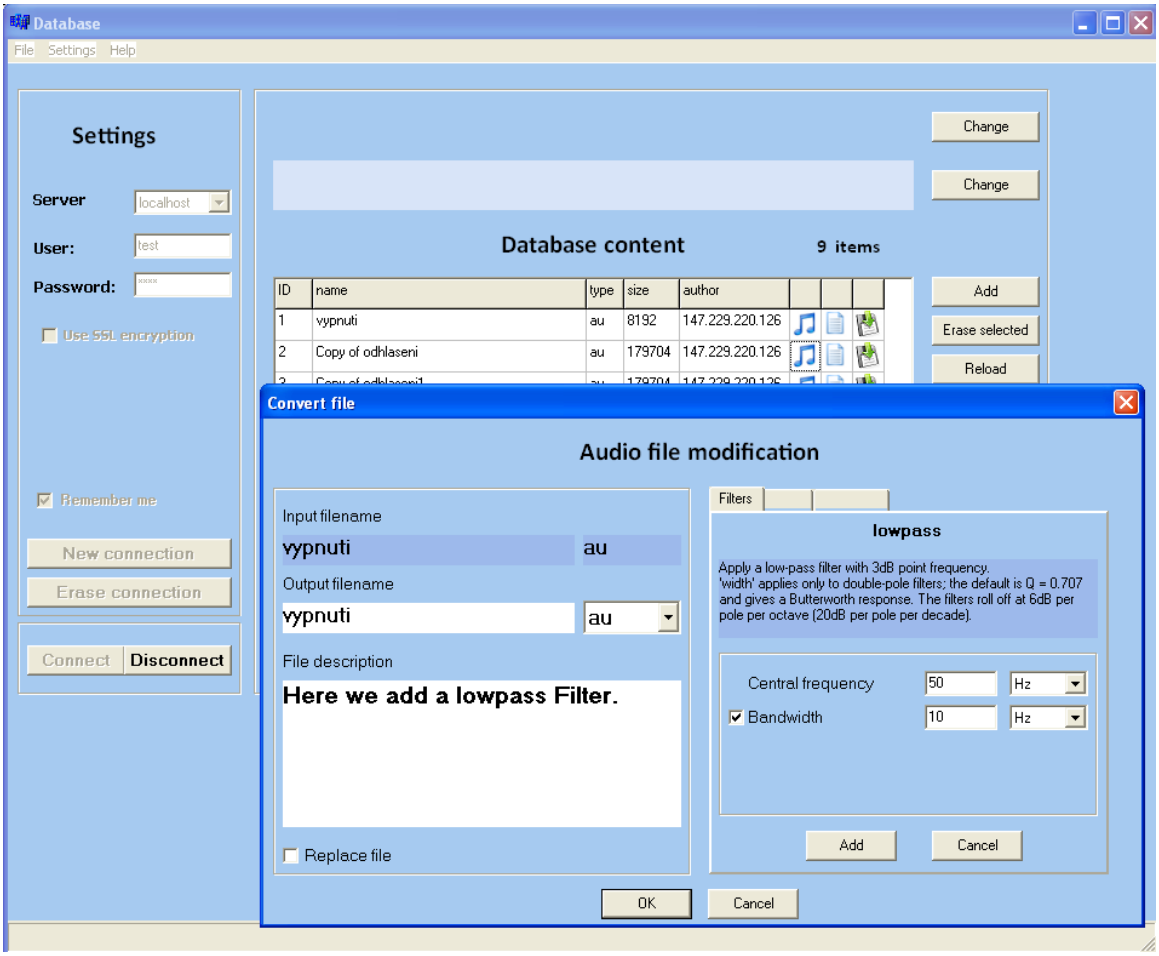


Figure 10.3 Filter Parametrs

and also more parametrs like gain in Equalizer which is measured in dB.

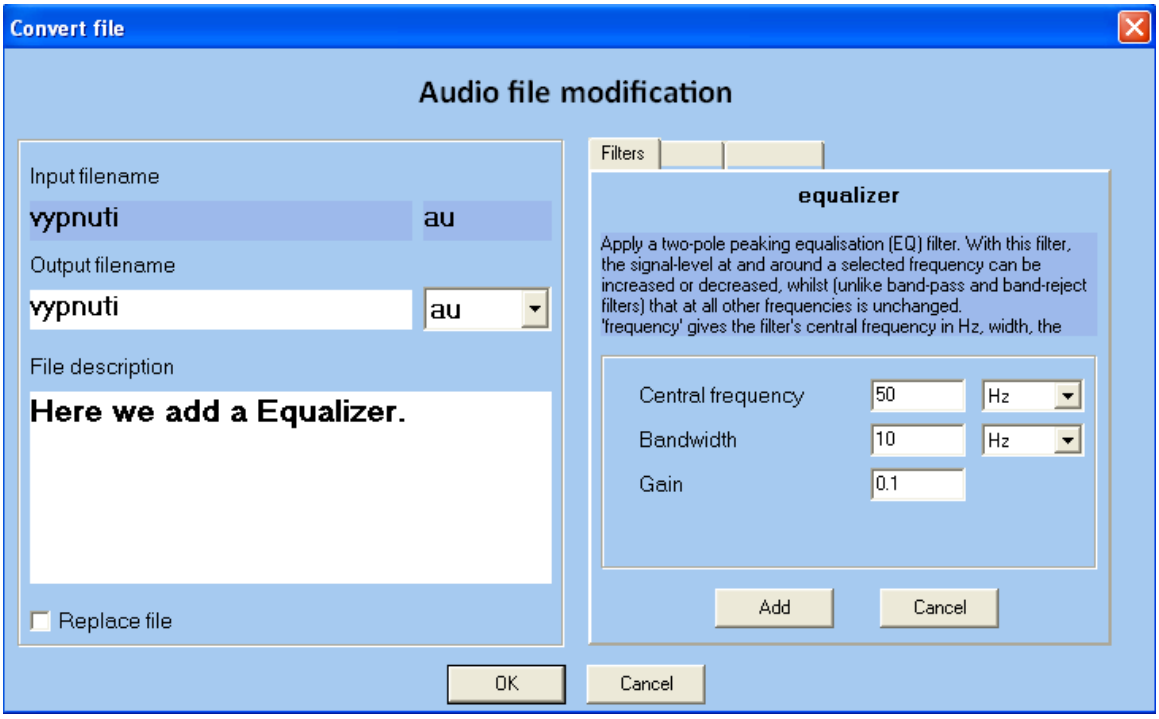


Figure 10.4 Equalizer Parametrs

11 Remove the noise by Sox

Virtually all audio recordings, no matter how they were created, will contain some amount of noise. Noise is simply any unwanted sound, from hiss, clicks, and static to a door slamming or someone coughing. Noise levels are typically higher in analog recordings, but some noise exists in all digital recordings as well.

The good news is that noise is only a problem if its level is above the threshold of hearing. If a small amount of noise is present, but at such a low level that no one can hear it, who cares? Well, some people might, but if you can't hear it, it's not really doing any harm.

Even noise above the threshold of hearing is not necessarily a problem if the noise is masked by louder material. For example, hiss, clicks, and pops that would be very annoying in a classical piano recording might be completely unnoticeable in a heavy metal song.

Distortion is similar to noise, but there is a key difference: Noise is something that's introduced to the signal from outside, such as hum, whereas distortion is a modification of the signal itself. Hit the Pause button and the distortion goes away, but the noise will remain. Another type of noise called thermal noise is generated by the electronic circuits themselves and can never be eliminated, though it can be minimized by the use of the higher quality components typically found in high-end sound cards and audio interfaces.

11.1 Types of Noise

Broadband noise, also called continuous noise, includes sounds such as hiss and static that span a wide range of frequencies. There's a certain amount of broadband noise in all recorded audio—even in recordings made by professional engineers with top-of-the-line equipment. The important measure here is the ratio of the average level of the program material to the average level of noise. This is called the signal-to-noise ratio (SNR) and is specified by the relative difference between those levels in Decibels (dB).

Narrowband noise is limited to a narrow range of frequencies—typically a fundamental frequency and its harmonics, or whole-number multiples. Examples include 60-cycle hum and the whine of an electric motor. A recording containing 60-cycle hum will likely also contain hum at 120Hz and 180Hz (two and three times the fundamental frequency, respectively). Hum is often caused by incorrect grounding and poorly shielded cables.

Impulse noises include brief, sharp sounds such as clicks and pops. Clicks are usually caused by small scratches or specks of dust on a record. They also crop up during digital recording when digital devices aren't synchronized. Pops are caused by more severe scratches. When you view a recording in a waveform editor, clicks and pops show up as steep spikes or dips (see Figure 3). Clicks are narrow spikes and may span just a few samples. (Each sample or data point at standard CD resolution is 1/44100 of a second.) Pops last longer than clicks, and can affect the signal for several dozen samples.

11.2 Noise Sources

There are many places where noise can get into an audio signal. Most are when the signal is in analog format. Following are some common sources of noise.
Original recording environment. When an original recording is made, noise will be picked up by any microphones and from the cables and circuits that handle the signal while it is in analog

form. Once the signal is captured or converted to a digital format, it will be fairly immune from noise unless you edit it or re-encode it.

Defects and wear of analog media. When you make a digital recording from analog media, such as a record or tape, noise will be picked up from the media when it's played, even if it is in good condition. Even more noise will be introduced if the media is worn, dirty, or damaged.

Analog equipment and cables. When you play an analog recording to capture it in digital format, noise will be picked up as the signal passes through the analog circuits of the tape deck or turntable, the preamp, and any interconnect cables.

Analog circuits in your sound card. A poorly shielded sound card will pick up noise as the signal passes through its analog circuits on the way to the digital-to-analog converter (DAC or A/D). The converter adds more noise from *quantization errors* as the voltage of the analog signal is sampled and rounded to whole numbers.

Errors during digital signal processing. Even when audio is in digital format, it's not totally immune to noise. When you edit digital audio you can easily introduce noise in the form of digital clicks and noise from quantization errors that are an unavoidable result of digital signal processing.

Sox uses filter ,the filter needs one argument, profile-file, which contains the noise profile from noiseprof. thershold specifies how much noise should be removed, and may be between 0 and 1 with a default of 0.5. Higher values will remove more noise but present a greater possibility of distorting the desired audio signal. Experiment with different threshold values to find the optimal one for your sample.

This sound remover is ideal for removing constant background noise such as fans, tape noise, or hums. It will not work very well for removing talking or music in the background.

Removing noise is a two-step process. In the first step, you select a portion of your sound which contains all noise and no signal, in other words, select the part that's silent except for the noise. Then choose Noise Removal... from the Effect menu and click Get Profile. Audacity learns from this selection what the noise sounds like, so it knows what to filter out later.

Then, select all of the audio where you want the noise removed from and choose Noise Removal... again. This time, click the "Remove Noise" button. It may take a few seconds or longer depending on how much you selected.

If too much or not enough noise was removed, you can Undo (from the Edit menu) and try Noise Removal... again with a different noise removal level. You don't have to get a new noise profile again if you think the first one was fine.

Removing noise usually results in some distortion. This is normal and there's virtually nothing you can do about it. When there's only a little bit of noise, and the signal (i.e. the voice or the music or whatever) is much louder than the noise, this effect works well and there's very little audible distortion. But when the noise is very loud, when the noise is variable, or when the signal is not much louder than the noise, then the result is often too distorted.

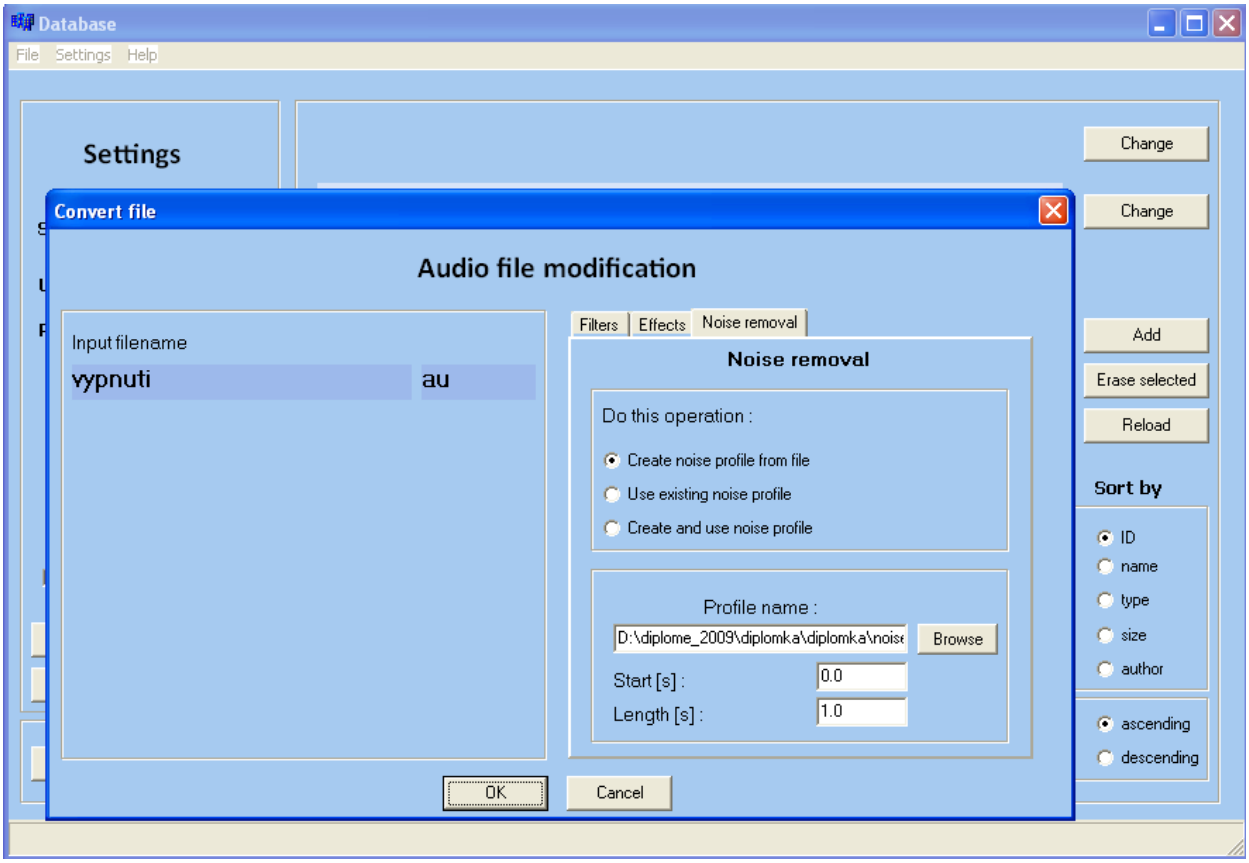


Figure 11.1 Noise removal Parametrs

and here we show how to create nosie profile by giving parametrs like the start of the file and the length,which are given in second.

11.3 FMOD Ex

FMOD EX is a programmer API which allows playing music and creates sound effects. The FMOD Ex sound system is a revolutionary audio engine for game developers, multimedia developers, sound designers, musicians and audio engineers. The development of FMOD Ex is based on the years of experienced of Firelight Technologies' previous product FMOD 3. FMOD Ex is intended to push the creative boundaries of audio implementation for games and the like, whilst using minimal resources and being fully scalable.

FMOD Ex provides both a low-level API and data-driven API (used in conjunction with Designer).

The FMOD mixing architecture starts with a fantastic sounding engine, that uses floating point calculations with full 32bit interpolation to provide maximum sound quality and headroom

when summing signals. Using a node based architecture, FMOD Ex provides flexible routing, submixing and output channel choices to the programmer. Input channels can be mapped to any output channel through a simple 2D matrix. Output to mono, stereo.

Included with the API is a whole suite of 14 DSP effects, such as echo, chorus, reverb, etc which can be applied throughout the DSP mixing network.

FMOD Ex includes advanced support for compressed sample formats such as mp2, mp3, ADPCM and XMA. Compressed samples can be looped or sequenced with other samples, without gaps, clicks or other artifacts. Developers do not have to worry about anything to do with the destination format, FMOD's encoder will just handle it perfectly for you every time!

Advanced streaming engine supports low cpu overhead, multiple stream support, over-ridable file callbacks and more.

FMOD supports a huge range of audio file formats including: wav, midi, mp3, XMA, ogg and mod just to name a few. You can even add your own codecs via FMOD Ex's plug-in support. FMOD Ex can play audio files with up to 16 channels.

Platform Support

FMOD Ex has the most largest range of supported hardware, including:

Windows (32bit and 64bit)

- Macintosh (PPC and x86)
- Linux (32bit and 64bit)
- Sony PS2, PS3 and PSP
- Microsoft Xbox and Xbox 360
- Nintendo Gamecube and Wii
- Solaris

11.4 Real-Time Transport Protocol

The real-time transport protocol (RTP) provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video or simulation data, over multicast or unicast network services. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services; both protocols contribute parts of the transport protocol functionality. However, RTP may be used with other suitable underlying network or transport protocols. RTP supports data transfer to multiple destinations using multicast distribution if provided by the underlying network.

RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence.

RTP consists of two closely-linked parts:

- The real-time transport protocol (RTP), to carry data that has real-time properties.
- The RTP control protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session. The latter aspect of RTCP may be sufficient for "loosely controlled" sessions, i.e., where there is no explicit membership control and set-up, but it is not necessarily intended to support all of an application's control communication requirements.
- V - Version. Identifies the RTP version.
- P - Padding. When set, the packet contains one or more additional padding octets at the end which are not part of the payload.
- X - Extension bit. When set, the fixed header is followed by exactly one header extension, with a defined format.
- CSRC count -Contains the number of CSRC identifiers that follow the fixed header.
- M - Marker. The interpretation of the marker is defined by a profile. It is intended to allow significant events such as frame boundaries to be marked in the packet stream.
- Payload type - Identifies the format of the RTP payload and determines its interpretation by the application. A profile specifies a default static mapping of payload type codes to payload formats. Additional payload type codes may be defined dynamically through non-RTP means.
- Sequence number - Increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.
- Timestamp - Reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations.
- SSRC - Synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC identifier.
- CSRC - Contributing source identifiers list. Identifies the contributing sources for the payload contained in this packet.

11.5 Real Time Streaming Protocol

The Real-Time Streaming Protocol (RTSP) establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video. RTSP does not typically deliver the continuous streams itself, although interleaving of the continuous media stream with the control stream is possible. In other words, RTSP acts as a "network remote control" for multimedia servers. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. RTSP is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as udp , multicast UDP and tcp , and provide a means for choosing delivery mechanisms bases upon RTP.

There is no notion of an RTSP connection; instead, a server maintains a session labeled by an identifier. An RTSP session is in no way tied to a transport-level connection such as a TCP connection. During an RTSP session, an RTSP client may open and close many reliable transport connections to the server to issue RTSP requests. Alternatively, it may use a connectionless transport protocol such as UDP.

The streams controlled by RTSP may use RTP, but the operation of RTSP does not depend on the transport mechanism used to carry continuous media. RTSP is intentionally similar in syntax and operation to HTTP/1.1 so that extension mechanisms to http can in most cases also be added to RTSP. However, RTSP differs in a number of important aspects from HTTP:

- RTSP introduces a number of new methods and has a different protocol identifier.
- An RTSP server needs to maintain state by default in almost all cases, as opposed to the stateless nature of HTTP.
- Both an RTSP server and client can issue requests.
- Data is carried out-of-band by a different protocol, in most cases.
- RTSP is defined to use ISO 10646 (UTF-8) rather than ISO 8859-1,consistent with current HTML internationalization efforts.

- The Request-URI always contains the absolute URI. Because of backward compatibility with a historical blunder, HTTP/1.1 carries only the absolute path in the request and puts the host name in a separate header field.

  The protocol supports the following operations:

  Retrieval of media from media server: The client can request a presentation description via HTTP or some other method.

- Invitation of a media server to a conference: A media server can be "invited" to join an existing conference, either to play back media into the presentation or to record all or a subset of the media in a presentation.
- Addition of media to an existing presentation: Particularly for live presentations, it is useful if the server can tell the client about additional media becoming available.

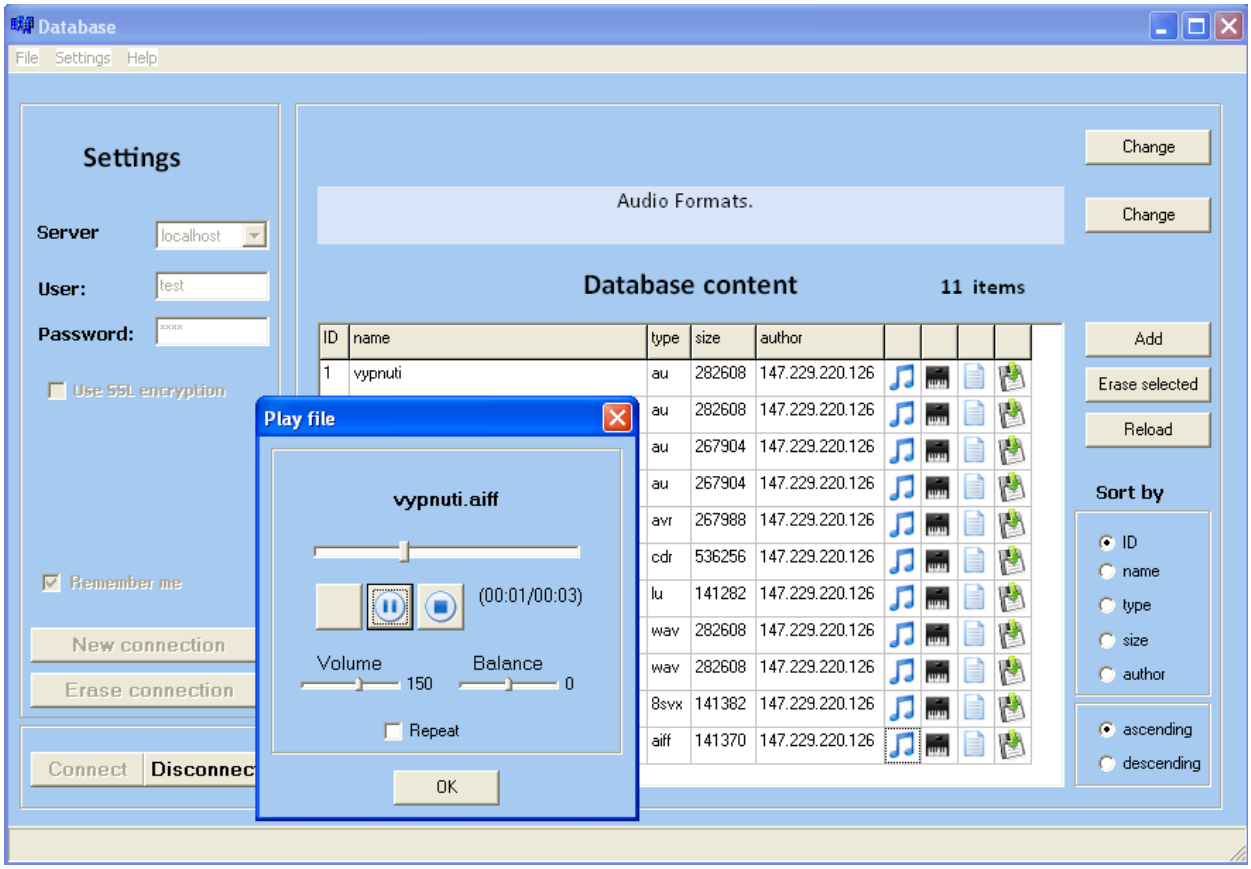


Figure 11.2 Play music by FMOD

by using FMOD Ex Library, which is free, so we could play our music files directly from our program, without the fading for download them.

Also by using this Library ,the program could change the volume and the balance of the sound, and also could repeat them.

Unfortunately FMOD doesnot support all formats which sox supports,thats why program can not play all music files directly from the program, which mean that should be downloaded then playing them.

As showed in the Figure, program does not support all types of format, like

AIFF,ASF,ASX,MP2,MP3 and so on.

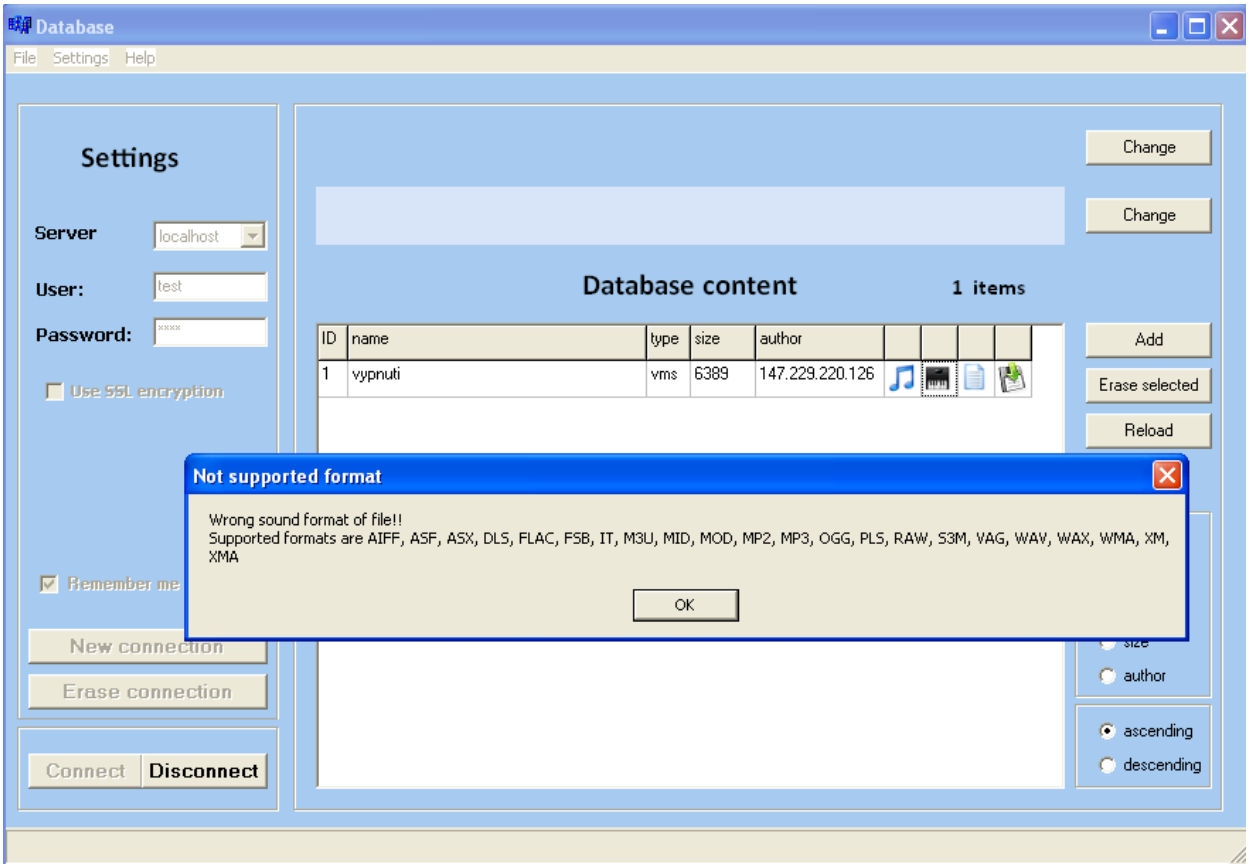


Figure 11.3 Supported formats by FMOD

12 Conclusion

In this project, we proposed effective, multi user and user friendly software written in C++ language for the remote administration of sound file server. The presented software enable the user to create, delete and modify each database, the bulk downloading and uploading of sound files is a matter of course. The login name and the password of the user which should be sent to the remote SQL server are secured using SSL protocol.

The system enables the user to remotely process the audio files stored on the server hard drive, for this purpose, the SOX library are used. The mentioned library contains powerful functions for editing and enhancing sound files: it enables the user for example to convert the format of the files in order to save space on the server hard drives.

There are some problems to handle in order to optimize the proposed system. For example, the number of expected users of the database should be taken into account in order to dimension the server needed, moreover, many commercial hosting don't allow the users to run applications on their servers, on the other hand, that ones which enable this feature, is financially consuming, however, it seems that the proposed system (the server and the administration utility) will be used for academic purposes, that means, the hosting server which can provide the sound file processing functions could be run on a server within the university network.

13 References

[1] Kitebird: The MySQL C API. 2007. www.kitebird.com/mysql-book/ch06-1ed.pdf (November 2008)

[2]MySQL:MySQL5.0 Reference Manual.2006. http://dev.mysql.com/doc/refman/5.0/en/c.html (December 2008)

[3] Matoušek, D.: C++ Builder vývojové prostředí - 1. díl. 1. vydání. BEN, Praha,2002. ISBN 80-7300-094-6

[4] http://sox.sourceforge.net/

[5] http://searchenterprisewan.techtarget.com/