

# **OPEN STREAMING MANUAL**

radioqualia  
October 2004

*Creating a live online radio station*  
version 0.1

**WHAT IS STREAMING?**  
**ENCODING AND DELIVERY CONCEPTS**  
**AN ANALOGY - TRANSMITTING AND STREAMING**  
**INTRODUCTION TO YOUR SOUND CARD**  
**EXTERNAL AUDIO DEVICES**  
**BASIC HARDWARE SET-UP FOR LIVE STREAMING**  
**MIXING DESKS**  
**CABLES AND CONNECTORS**  
**PLUGGING IN**  
**THE ART OF MIXING**  
**INTRODUCTION TO LINUX WINDOWING SYSTEMS**  
**INTRODUCTION TO LINUX FILE STRUCTURE**  
**SOME LINUX COMMANDS YOU SHOULD KNOW**  
**CONNECTING TO THE INTERNET WITH LINUX**  
**INSTALLING SOFTWARE ON LINUX**  
**INTRODUCTION TO LINUX SOUND ARCHITECTURE**  
**LINUX SOUND SOFTWARE YOU NEED TO KNOW**  
- MIXERS  
- XMMS  
- TEXT EDITORS  
- FTP  
**INSTALLING MuSE**  
**LIVE STREAMING WITH MuSE**  
**PUTTING IT ONLINE**  
- CREATING PLAYLIST FILES  
- LINKING PLAYLISTS FROM A WEBPAGE  
**THE POLITICS OF STREAMING : OGG vs MP3**  
**SETTING UP A STREAMING SERVER - ICECAST**  
**USING ICECAST**  
**STREAMING LIVE FROM PLAYLISTS WITH MuSE**  
**OTHER TOOLS**

next version....  
ADVANCED USAGE OF MuSE  
PUTTING ICECAST ONLINE  
SOME CASE STUDIES / ANECDOTES  
NEED TO KNOWS

## FOREWORD

Hoi...this manual is intended to assist anyone to set up a live streaming radio station under Linux. It is intended as a hands-on guide, so it would help if you had a Linux machine you can play with while reading this manual. Being online isn't absolutely necessary but it will help.

This manual is written by Adam Hyde of r a d i o q u a l i a (<http://www.radioqualia.net>) to assist in the many workshops on this topic that he does. At the same time it can be used without Adam(!). It is written in an extremely informal way, so no complaints about the grammar unless its *really* shocking!

Its free. Distribute as you wish for non-commercial purposes. Licenced as a Creative Commons Attributive Non-Commercial ShareAlike document. First released in this form in October 2004. Any offers to print it in manual form are gratefully accepted.

This is version 0.1 of the 'Open Streaming Manual - Creating a Live Online Radio Station'.

be good

adam  
adam@xs4all.nl

## WHAT IS STREAMING MEDIA?

Streaming media is the term used to describe the **real-time delivery** of moving images, moving text and sound, over the internet. Streaming software allows internet audiences to listen or watch types of media, which have, up until recently, been considered too large and bulky for consumption over the internet.

Streaming media techniques work in the following way: as you listen or watch one portion of content, the next portion is downloading at the same time. The ability to **simultaneously load and play** distinguishes webcasting from other types of internet media. Streaming allows for **live transmission** over the internet, which enables a transformation of the internet into a broadcasting medium.

Content can also be saved and archived in streaming formats, allowing internet users to experience recordings of live events online after they happen.

There are three types of delivery of audio and video data over the internet: **Download and Play**, **Progressive Download**, and **True Streaming**.

### Download and Play

- is the process whereby a user must first download the entire media file before playing it. Download and Play media cannot be used for live broadcasts, however it is often a good way to deliver high quality media content over any bandwidth (although download time may be a problem when delivering large files over lower bandwidths). MP3 is the most commonly used Download and Play audio media on the internet and DivX ;-) is becoming increasingly popular as a Download and Play video format.

### Progressive Download

- enables viewers to experience media as it downloads to their harddrive. Progressive Download is useful but is "unintelligent", and hence cannot provide advanced functionality such as multiple bitrate streaming (see below).

### True Streaming

- enables the user to view or listen to the media as it is delivered over the internet in real-time. True streaming particularly distinguishes itself in the way the client can control the media experience by pausing, skipping forward to a scene, or rewinding. Although different terms have been used to describe true streaming including, webcasting, livecasting, or net.radio the term that has prevailed is 'streaming media'.

NB : Progressive download is not streaming media although the difference between *Progressive Download* and *True Streaming* is at times ambiguous. If, for example, a user has a very fast connection to the internet a progressive download file may appear to behave in exactly the same way as a streaming file. The main difference between Progressive Download and True Streaming is that the latter has a more complex client-server relationship which enables more advanced functionality.

Within true streaming (which I will from now on refer to as *streaming media*) there are two distinct varieties : **static file streaming** and **live streaming**.

### Static File Streaming

- the delivery of pre-recorded media files over the internet in real time. Typically when we refer to archives of online media we are discussing static file streaming. A large archive of video art encoded into streaming files is an example of a static file streaming archive. This content is also known as *on-demand* content.

### Live Streaming

- the delivery of live audio and/or video over the internet, allowing the user to experience an event live (as it occurs in realtime). There are many examples of this such as online radio or viewing live performances.

This manual will mainly deal with live streaming. Although live streaming can occur in any number of contexts, the basics are the same, so we will look at how to set up a basic online 'radio station'.



## ENCODING AND DELIVERY CONCEPTS

To stream static files over the internet the files first have to be encoded into an appropriate streaming format. This is done with the various encoding tools (softwares) created by the main technology providers. In simple terms a digital file, for example a CD audio recording, is converted by the encoding software from its original digital audio format into a streaming file. This will involve compressing the data, which will reduce its quality and file size, and converting the data into propriety or open streaming formats. Streaming static files can also be captured ('recorded') from live audio / video inputs.

There are two forms of compression – *lossy*, and *lossless* compression.

When the encoding process compresses the source file so that it can be delivered over slow internet connections in real-time this also degrades the quality of the audio and video. The more a file is compressed, the lower bandwidth required to be able to play the file, but the more the quality is reduced. A compromise has to be achieved whereby the level of compression achieves a qualitatively acceptable audio and video experience, while allowing the delivery of the content over the internet connections that your target audience will typically have.

Live encoding is similar except that an audio or video (or both) input is encoded instead of a file. With this process the encoding software delivers the encoded data in a continuous stream to the streaming server (see below).

*Multiple bitrate encoding* allows the server and player to negotiate the best quality (highest bandwidth) stream to be delivered from a single static file or live stream. Hence the player is delivered the best quality stream possible over the users internet connection. Multiple bitrate encoding produces only one encoded stream.

Choosing the bitrate(s) will require the consideration of several factors, including (not exclusively):

- **The target audiences connectivity**
- The desirable frame size of the video (if including video)
- The amount of movement in the frame (if including video)
- The level of video contrast (if including video)
- The type of audio encoded (e.g. ambient noise/voice/stereo music)
- The amount of camera movement (if including video)
- The quality of the camera and camera lens (if including video)

### Lossy Compression

This is the process of discarding data when compressing a file so that the final file is different from the source file. The algorithms used are complex and each streaming architecture uses their own codecs to perform this operation. The effectiveness of each codec is evaluated by how intelligible the final media file is after removing the data. MP3, for example, attempts to remove data from the source file which represents frequencies that the user will probably not normally be able to hear. This process is called 'perceptual encoding' because the compression algorithm tries to preserve the qualitative perceptual experience of the user as much as possible when data is removed.

### Lossless Compression

This is the process of compressing data information into a smaller size without removing data. To visualise this process imagine a plastic bag with an object in it. When you remove the air in the bag by creating a vacuum the object in the bag is not effected while the total size of the bag is reduced.

### Delivery

Streaming static ('archived') files can be achieved using a normal *web server*. This is often the cheapest way to deliver content on a small scale and is known as *psuedo-streaming*. This method cannot be used for live streaming and does not allow for the advanced features of true streaming such as multiple bitrate encoding. Psuedo-streaming will also enhance the likelihood for time-outs ('buffering') and cannot deliver the same amount of simultaneous player connections as true streaming.

To enable live streaming and to gain the full functionality of streaming static files, a **streaming media server** is generally (but not always as sometimes a server is not required in some instances when the encoding software itself can also as a server) required. For static files, these are hosted from a specialized streaming server connected to the internet. This server is usually standard server hardware but with the necessary streaming server softwares installed. It is quite normal to install a streaming server on the same machine as an existing web server. The most popular server softwares are still the commercial products produced by Realnetworks (RealServer), Apple (Quicktime Darwin Server) and Microsoft (WindowsMedia server). Less prolific but rising quickly in popularity are open source servers including Iccast and Jroar. QuickTime Darwin is now a certified 'Open Source' server software.

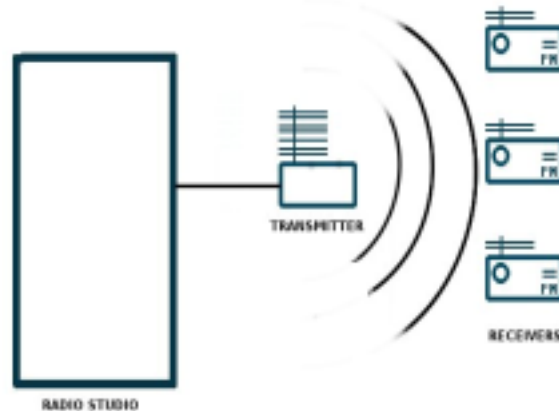
Placing a static streaming file on a streaming server is the same process as putting a web page on a web server (using FTP applications). The media player (for example the RealPlayer or XMMS) then connects to the content via the streaming server and hence true streaming is realized.

To deliver a live stream the same type of streaming server is required (a single streaming server can be used for both live and static file delivery). However the live stream delivery process is quite different. In this process the encoding software delivers a continuous stream from a computer to the streaming server. This encoding computer will typically have a soundcard and video card installed and will encode these inputs and deliver this in a continuous stream to the streaming server.

Similarly with the static file streaming, the player connects to the content via the streaming server.

## AN ANALOGY - TRANSMITTING AND STREAMING

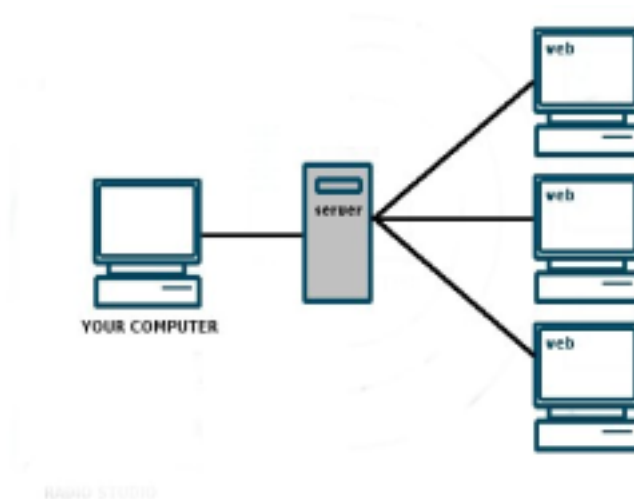
Perhaps a good way to understand how streaming works is to imagine the situation of a radio station. A radio station comprises of three components - a studio, a transmitter, and the receivers that your audience has.



Above is a basic diagram showing how a transmitting radio station works. The radio studio is the source of the audio. In this space there are usually mixing desks, cd-players, minidisc players, turntables etc. Then from the studio an audio signal is sent to the transmitter. This can be sent from the studio to the transmitter by either a cable (sometimes called a 'landline') or by a microwave link. Then the transmitter sends the audio via FM so that radio receivers (tuners) can pick it up and play it.

Radio works this way because it is trying to distribute the studio audio to as many people as possible. If you imagine the radio studio without the transmitter then the station would have a fairly reduced audience! Essentially only people that could fit into the studio would be able to listen. So the transmitter works as a distributor, allowing more people to connect via their radio receivers and hence the potential audience is enlarged.

A similar motivation and process exists for streaming. If you were just playing audio in your room then the audience isn't going to be so big...so, we utilise streaming to distribute the audio to more people.



The analogy is obvious....the computer replaces the radio studio, the streaming server replaces the transmitter, and your listeners connect by computers to the server rather than with radio receivers to the broadcast signal. The analogy can be taken quite a long way. Having a bigger radio transmitter is like having more bandwidth available at the streaming server - both allow more people to connect.

The question you might ask is...why don't the listeners just connect directly to my computer? Well, its a good question..it is absolutely possible to consider doing this. However it is often true that you don't have enough bandwidth available where your computer is located. For example, if I was wanting to send out a low-bitrate stream ....lets say, 56kbps - this is the standard speed of a dial-up modem. Well, if I was connected to the internet with a dial-up modem and I wanted to send a 56kbps stream, then if

someone connected to my computer directly to listen to the stream then all my bandwidth is used up. No one else would be able to make a connection so I would have just 1 listener. If however I send my audio to a streaming server, then usually this server is placed some where like at an Internet Service Provider (ISP) which has much more bandwidth than I do - hence more people can connect.

The next question is ...what then is the difference between my computer and a server? Well, there isn't necessarily any difference. We tend to think of 'servers' as monolithic machines capable of taking the worlds internet demands on their shoulders...but actually, technically, a 'server' is just a computer, like the one you have, which runs specific software. These softwares are called 'services' or sometimes they are called 'daemons'...so if a machine is running a 'streaming daemon' it means that it is acting as a 'streaming server'. For example, if you run the **Icecast** software that is covered later in this manual, then your machine *is* a streaming server. Your machine can do this just as easily as the (admittedly powerful) machines at the Internet Service Provider.

It is even possible to run the encoding software, and the streaming server software *on the same machine*....

## INTRODUCTION TO YOUR SOUND CARD

On the most basic level, a sound card is the ears and mouth of your computer - allowing the computer to 'listen to' (input) and make (output) sounds. The sound card can accept sound inputs like microphones, mini-discs, external music players (mp3 players, CD players etc) so you can listen to, or record this sound. Additionally the soundcard can output audio from your computer to a set of speakers or headphones.

Most commonly the sound card on a desktop computer accepts one or two different types of audio input, : Line-in and microphone (Mic). The inputs on most computers these days are marked by colors and symbols, and generally they are as follows:



The two connecting ports above are the line-in (blue) and the Mic-in (red/pink). These icons can vary between computer or soundcard manufacturers, of course, as there is no standardised representation of sound card ports. Sometimes for example, on laptops there will be only two ports, and often without icons but simply using the color codes of red (Mic-in), and green (headphone/speaker out).

Another common way of indicating which is which is by etching symbols on to the metal, and in this case the following are often used:



In anycase you will generally attach your CD-player or minidisc etc here via a 3.5 mm jack (mini-jack) such as this:



But more on this later...

## EXTERNAL AUDIO DEVICES

The first obvious question is...what is the point of having two different types of input? Well...without wanting to get into too much sound theory...

Microphones generally generate a very low powered signal, and hence when they are connected to a sound card they need more amplification than something like a CD player or MP3 player (which generally have their own built-in amplifiers). So a Mic-in on a soundcard has a dedicated amplifier, whereas the line-in does not.

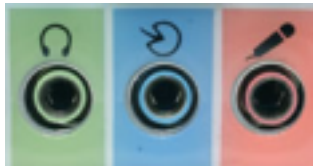
You can use the line-in for your microphone but you will need to supply your own amplifier to increase the output of the microphone to a level that is acceptable to the line-in. This can be a good idea if you have a better amplifier available than the one which is built into your soundcard. Never plug a CD-player or similar into your microphone input as you run the risk of overloading and burning out your soundcard.

The "Line-in" input requires a higher input signal level than does the "Mic" input. In other words, the Mic input includes an amplifier that the "Line-in" input does not have. In order for you to use the "Line-in" input you must provide your own amplifier in order to increase the output level of your microphone to a level compatible with the "Line-in" input. If your amplifier is less noisy or contributes less distortion than the one included on the "Mic" input you will, indeed, have a higher quality signal. If your amplifier is noisier or less linear, you will end up with a lower quality signal.

So, in general the following list applies to typical devices:

Line-in	Mic-in
CD player Mini disc tape deck transistor radio mixing desk	Microphone!

....alright...so the other component to your soundcard will be your output port, in the following diagram this is the green box on the left:



The headphone symbol doesn't mean you can only plug headphones into this output, it means you can connect headphones or speakers...its just that a headphone is easier to represent in icons than a pair of speakers.

## MIXING DESKS

In a Live Streaming set-up you are generally wanting to combine several live inputs into one live stream. Lets say, for example (and this is one of many scenarios) that you want to set -up something like a small online 'radio station' with a studio similar to a what a radio studio has. So, as a basic suggestion you could have the following audio devices:

two CD-players  
minidisc  
microphone

so...that would be a pretty basic studio. The question is...how do you get all these inputs simultaneously into the soundcard? Well, when I ran a small community TV station in New Zealand (StaticTV, now known as BigTV) we decided to do it the hard way...basically, every time we would want to change from playing one video program to playing the next, we would quickly pull out the cables from the transmitter that connected one video player, and quickly plug in the cables from another video player! Not the most elegant, but it worked...however an easier way would have been to use a video mixer. Unfortunately video mixers are expensive, so we couldn't afford one, but fortunately for you, audio hardware is much cheaper than video hardware. So in this situation I suggest you take the easy road, and look at plugging an audio mixing desk into the sound card and then connecting all your audio devices into the mixer.



The above image is a 4 channel mixer...if you are starting to get worried about this looking a bit complex, then don't be...the basic concept is simple....you plug your audio devices (CD-players, Mics, minidisks) into the mixer, and the mixer controls the volume of each device...simple....So, all you really need to know is where on the desk you need to plug in the CD-players (etc) and then how to plug the output of the mixer into the line-in of the soundcard.

This isn't too hard as all mixers are essentially the same...the biggest difference between mixers is how many inputs (called 'channels') it has. In the example above there are 4 channels, meaning that you can input 4 audio devices.

In the example above you can see four columns of buttons going from top to bottom - these are the buttons controlling the audio for each of the four channels. Each vertical column of buttons is a separate channel, with a corresponding input jack at the top of each column. Your audio plugs into the channel inputs, and you plug the 'main output' into the soundcard...

Its possible to do this in other ways if you have multiple audio devices and no mixer...the most basic way is the way I described above (pulling cables in and out of the line-in port)....but this will make some crunchy sounds as you pull cables in and push them in...you could manage this by carefully turning the input volumed down using software (more on this later) but, lets face it, if you can avoid this situation then its probably for the best...however, you could try combing audio inputs using a 'splitter' connector....ok, to be honest, i'm not sure if thats the technical name for the component I mean...I always call it a 'splitter' and no-one has ever complained, and it looks like this:



So you can take two mini-jack inputs with this and combine them into one input for your soundcard like so:



if you do it this way, you can use the audio-in to input two (or more?) devices and you can 'mix' the audio just by using the volume on the audio equipment itself (CD Walkmans, for example always have a volume control)...this isn't a bad solution and I have done plenty of streaming this way (especially in the early days of the XCHANGE network - <http://xchange.re-lab.net> - when no-one could afford a mixer...)...this method works but you have to be prepared to accept the occasional bad mix if you forget to turn something down etc as you will be juggling several tasks at once...

the other way is to switch between audio devices by using a switch such as this RCA switch:



These are cheaper than mixing desks, but they don't allow you to 'mix' the audio volumes between devices...it allows you to choose an audio input device and then switch to another one

## CABLES AND CONNECTORS

Cables and connectors are probably one of the most important pieces of technology you will use in a stream set-up...this might seem over-blown, but its true to the point that you can have all the fancy input devices you want...DAT players, microphones, satellite receivers...but if you don't have that cheapo connector you were meant to buy the day before yesterday, then you won't get the signal into the soundcard and hence the show doesn't go on...

In the basic streaming set-up there are three kind of cables that are essential:

### 1. mini jack



The mini jack is also known as a 3.5 millimetre jack. get your hands on a few of cables (the cables have the same adapters on each end). The above example is a mono version, below is a stereo version:



You'll notice that the difference is in the number of black lines...look for this when you buy the cables, 1 black line = mono, 2 black lines = stereo. You will probably need more stereo than mono, as most audio devices you need have stereo outputs.

### 2. RCA Cables

Also known as 'Tulip' or 'cinch' cables. Some audio equipment will have mini-jack outputs, most likely this will be portable devices like walkmans etc, other equipment (like domestic CD-players, DVD players etc) will have RCA outputs.



If you can get some of these cables it could amke your life a lot easier. The above is a stereo RCA cable, if you split the cable in two you end up with two mono RCA cables.

AN invaluable hybrid of the two above, is a mini-jack to stereo RCA cable...make sure you have some of these if you can.



They don't need to be as long as this example...usually i think most cables in this variety come at about 0.5metre in length or so.

### 3. Microphone cables

microphones can come in all shapes and sizes, and they can have varying types of output connectors. Small microphones like the type you might buy for a domestic video camera, or a mini-disc recorder, will usually have their own cable attached and it will most likely be a mini-jack output at the end. However the more 'professional' versions, will require a microphone cable to connect the microphone to the mixing desk. If you have one of these microphones then make sure you get a Mic cable for it too.





In addition you should try and get hold of some connectors, these are the for saving you when you don't have the right cable for the right job...essentially if you have a few RCA cables, a few mini-jack cables and some connectors, then you can save the world...

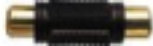
Look for the following:

#### **RCA-mini**



For connecting RCA cables to a mini-jack input.

#### **RCA Barrel**



For extending RCA cables by joining them together.

#### **Mini-jack to jack converter**

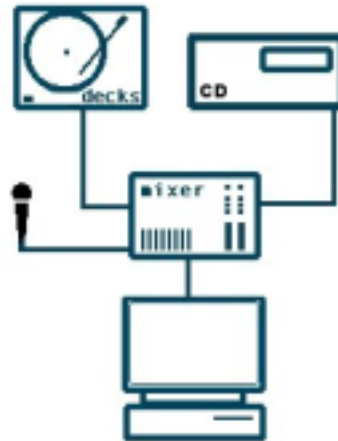
For converting mini jack cables to the larger 'jack', also sometimes called a 'headphone jack'. Additionally, a RCA-Headphone jack converter is almost always necessary (we will use them in the example below)..

Actually my recommendation is - if you are lucky enough to live near somewhere that sells these kind of connectors, get all the spare change you can muster, and buy every different type available....you won't be disappointed, and that moment when everyone is running around trying to find a BNC to 3.5 mono-jack cable - you won't have to buy a drink all night :)

## BASIC HARDWARE SET-UP FOR LIVE STREAMING

So...now to plugging it all in....for this I am just going to suppose you want to set up a 'radio station' for streaming online...whether you have a transmitter beside the point, but if you want you should build or get a transmitter! For examples on how to build your own transmitter you could look at Tetsuo Kogawas Site ( ) and at a r a d i o q u a l i a project (done in collaboration with Jan Gerber and Thing.net) which combined wireless internet (802.11b), streaming, and fm radio : <http://fm.thing.net>

But, for now....the online radio station....we have already gone through all the basic hardware elements....now how do we plug them all in? Ok...it should look something like this:



Images above are taken from the excellent Irrational guide to net.radio: [http://www.irational.org/radio/radio\\_guide/](http://www.irational.org/radio/radio_guide/)

In the image above we have several core elements....they are - the audio sources (decks, CD-players, microphone) , the mixing desk, the computer. Now it is possible of course to have less equipment than this....infact you could do a live radio station by throwing it all out, except err...maybe keep the computer (or use some paper cups and string ;)....at one point, when desperate a friend (Peter Kirk) and I sent a live stream from Australia to Austria streaming by plugging some headphones into the mic-in port and shouting into the headphones... sound crazy? well...yes...but it worked as headphones are essentially microphones in reverse...if you move the diaphragm in a headphone it generates a current, which is just how a microphone works....the point being - you can improvise..I have also at times streamed from laptops and used the built in mic (sometimes the mic-in port on laptop soundcards also double as actual microphones)...but it has to be said - others might not think sitting in a room talking to your laptop is completely sane...sometimes at least, external audio devices lend you the appearance of sanity (haha).

So, don't get depressed at the array of equipment i have suggested...also, to stream you don't need a hugely fast machine...i don't know what the bottom line is, but (once again with an anecdote) when streaming a project (free radio linux - <http://www.radioqualia.net/freeradiolinux>) for a year and a bit and I used a beat-up Pentium II 166Mhz machine and it was extremely stable...remember too that unless you are doing archiving on your local harddrive that you don't need much harddrive space...so streaming audio under linux could be as little as a Pentium II 166Mhz machine with, say, 2 GB disk space (big enough to fit the operating system and streaming software) ...to further open the possibilities, in some situations, if you don't need external audio inputs (say, if you just want to stream mp3s from your harddrive) you don't even need a soundcard...

so...start thinking about what you need for your situation....don't get all technofetishist (unless you like it and can afford it)...its good to start with minimal equipment and build upwards as it helps liberate the understanding of what you are doing from the technical environment, and that can be very useful when you come to trouble shooting or inventing new projects...

## PLUGGING IN

so...lets go with the above set-up and look at how we plug it all in....

now...the check list (for the above scenario):

1. encoding computer with soundcard
2. mini-jack to mini-jack cables
3. RCA-RCA stereo cables
4. Microphone
5. Microphone Cable
6. Mixing Desk
7. Assorted connectors
8. Assorted cables
9. CD player
10. Turntable

so...first up...set up on a desk somewhere with everything handy, make sure you have enough plugs for the power of all the devices, and that you have everything close together so that the cables will reach (you would think i wouldn't have to say that ;)...)

Lets plug in everything one by one...first up we'll plug the microphone into the desk...now although I said all desks are created equal, its not quite true as some desks have different types of input sockets...in the desk I have pictured above (4 channel Berringer Desk) there are two sockets for microphone cables...one socket is on channel one, and the other is on channel two..You will need a microphone cable that goes from XLR (the name of the microphone jack) to XLR ...and XLR has three pins as so:

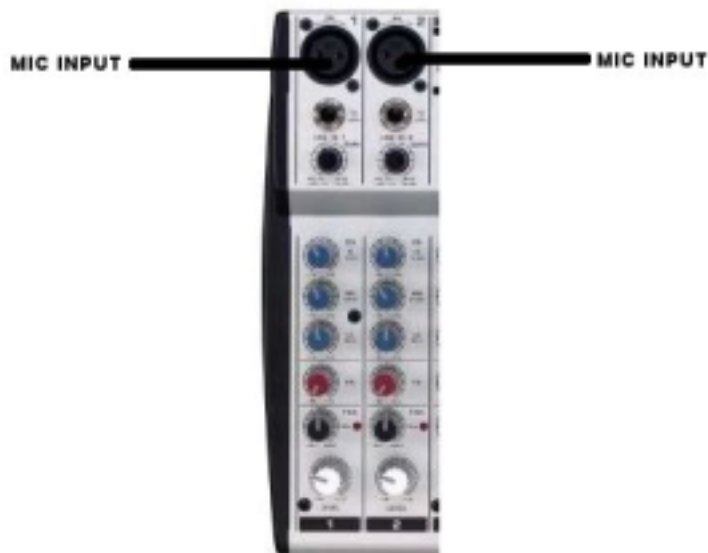


these type of connectors are also known as 'canon'. There is a standard way of describing each end of the cable...one end is known as the 'male' the other the 'female'...i let you work out which is which ;) ... this standard way of describing cable connectors (male/female) applies to all connector types not just XLR.



It could be that the input on the desk you use for a microphone accepts a 'headphone jack' type connector in which case you will need a XLR to Headphone-Jack cable. It is a very common mistake to get the wrong cables to connect devices to the mixing desk (or to each other) so if necessary take your equipment to the cable shop and lay it out on the counter and try it all out...don't be shy...

To plug the mic in, first attach the microphone to the microphone cable...then plug in the other end of the cable to the desk...in the example mixer we use you will see the mic input here:

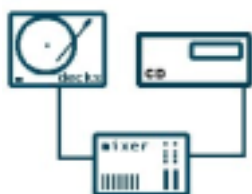


This will look different according to what kind of mixer you will be using but the principle is the same. It doesn't matter which channel you use...if you don't have a desk that accepts mic inputs then you might have a bit of a problem because the microphone needs amplification (see the section above about mic inputs vs line inputs)...if this is the case (you don't have a desk with a microphone input then you will need to either buy a microphone amplifier, or you could revert to plugging the microphone directly into the mic-in of your soundcard. This will mean you have to balance two inputs with your software (more on this later) but actually, it's not so bad...this issue will certainly be the case if you are using a RCA switch as I described above as the RCA switches never have amplified microphone inputs

If you do decide to plug your microphone directly into the soundcard make sure you have something to convert the microphone cable to a mini-jack....

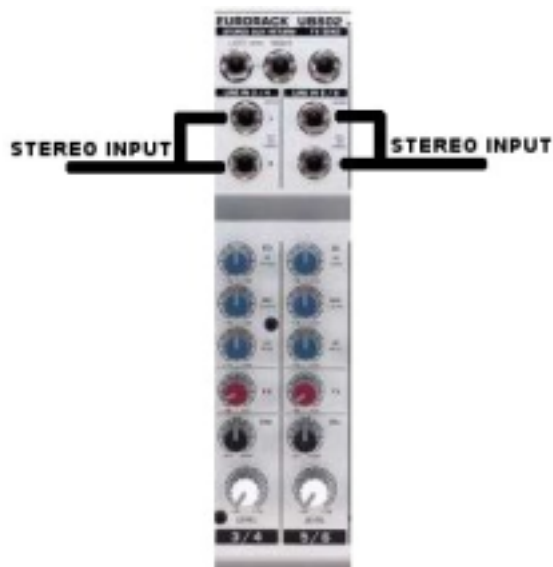
Another way to solve this problem would be to plug the microphone into another audio device that is capable of recording...for example a portable recording minidisc player - sometimes these devices allow you to monitor the audio in (microphone) and hence you could use the device as an intermediary amplifier. In this situation you would connect the mic to the recording device and then take the line-out or headphone out of the minidisc and connect it to the mixing desk. Sometimes you will need to press 'record' or 'record' and 'pause' on the recording device to get a signal.

Next up...the other external audio devices, in this case the CD-Player and the deck(s).



Essentially, these are the same as you will most probably have RCA connectors at the back of both of these types of devices. Usually turntables have a cable already attached with two RCA connectors, while CD-players (unless they are walkmans in which case they use 3.5mm mini-jacks) will have two RCA output sockets at the back on the device.

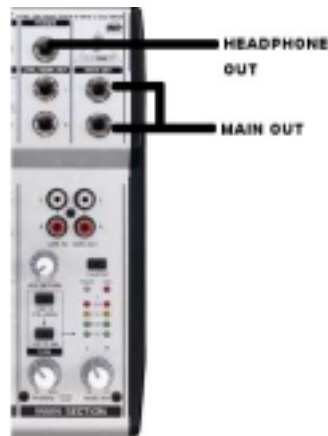
So...here is where you will need your RCA cables (stereo) and also, you will need some RCA-Headphone jack converters. Now plug in the RCA cable to the back of your CD-player, and then connect the other end of the RCA cable to the desk...Here is a tricky bit - mixing desks have mono and stereo input...you will need to know which channels on your desk (if any) are stereo inputs. Most likely they will all be mono inputs, but in the example desk we are using the last two channels are stereo. This means I could either connect the RCA cables coming from the CD-player to two mono inputs on the desk (channels 1 and 2 in this example) or I could connect the CD-player to just a single stereo channel..



If you decide to connect the CD-player to a single stereo input, then you will need to get a connector that converts stereo RCA to a stereo headphone jack. This choice can be an advantage because it means you just use one volume control to control the overall output volume of the CD-player. If you choose the second option and connect the CD-player to 2 mono channels then when you alter the volume of the CD-player on the desk you will have to do this simultaneously on both channels...this is not usually a problem but you will have to remember that you set it up this way.....Often to help remember what channel has what input on a mixing desk sound engineers use masking tape to put across the very bottom edge of the mixing desk below the columns of channel buttons and write the name of the audio source with a marker pen.



Now you are ready to connect your mixer to your computer. This should be done utilising the 'main out' connectors...if you don't have something on your desk that is labeled 'main out' then it might be called 'speakers' or some such...in a worst case scenario and you can't work out what the appropriate output socket is, you can also use the 'headphone out' socket which most small mixing desks use.

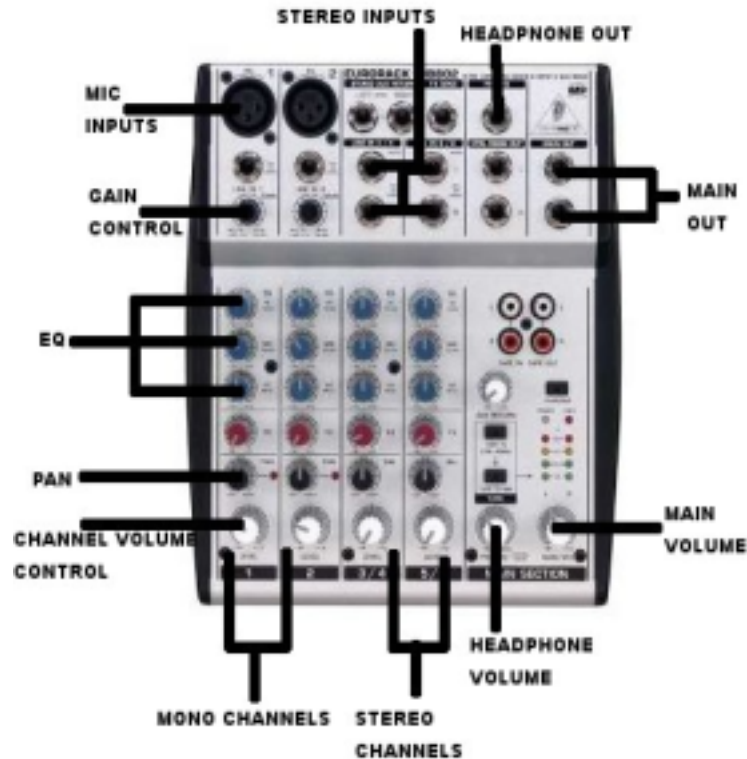


Then connect the other end of the cables to the computer through the line-in.

Thats it! phew!

## THE ART OF MIXING

Before you get streaming, its a good idea to first get used to mixing....this is really an art and a profession, so first thing to do is break all the rules...plug in your headphone or perhaps plug in some speakers, and start jamming with some audio devices...get used to your desk and how it works and try twiddling all those buttons that you don't understand...when you have done this for a while you might want to look a little of the following



OK! now we are getting somewhere!....so the basics of the mixing desk are, as per heading:

### MONO CHANNELS

As you can see, on this desk there are two mono channels. When we say 'channel' we mean everything associated with a specific input on the desk. So if I was to say "the mono channel" I essentially mean the controls that effect a specific input, that is all the knobs in the column under the audio socket. So "channel one" refers to the column of buttons and on the far left, "channel two" is the next column of buttons etc....Channel 1 and 2 on this desk are mono channels...they are also the channels where you can decide if you want a mic or a line input.

### STEREO CHANNELS

These are the same as mono channels, except they accept stereo inputs instead of mono inputs. It is always a good idea that you double check what connectors you have going into the various mono and stereo input sockets before you start streaming. Plugging a mono jack, for example, into a stereo socket will only give you only one half of the stereo input which usually means you are only getting one side of a stereo audio input. Plugging a stereo jack into a mono socket sometimes just doesn't work...

### CHANNEL VOLUME CONTROL

This is one of the most important parts of a mixing desk. The channel volume controls is the thing that...controls the volume on that channel (tautology?)...so when you first plug in a device *make sure the channel volume is set to zero* ... this is very important because otherwise you might hear a loud crunchy noise which can be particularly unpleasant if you are wearing headphones.

Some desks have a slider instead of a button, infact a slider is more common....

but as you can see, all desks follow the same principle whether they are hardware, software, 4 channel, 64 channel, with sliders or with knobs...

### **MAIN VOLUME**

The main volume control on the example desk is also a knob and it is also pretty important...if it is not turned up, then none of the channels, no matter how high you have set their individual volumes, will be heard...the main volume is also sometimes called the 'master volume'. This is the component that controls the overall volume and it controls the level of output volume that goes into your soundcard. Balancing the channel volumes and master volume to get the sound just right is a large part of the art of mixing.

### **MAIN OUT**

As discussed earlier, this is where you plug in either a set of speakers (which may need an amplifier) or it is where you take the output which will plug into your soundcard.

### **HEADPHONE OUT**

Where you plug in your headphones so you can simultaneously monitor what is happening. Note that this is not a redundant activity...sometimes you are not able to hear clearly the output from the main out, for example when streaming you might not have good speakers attached to your computer so you cant hear clearly the signal coming from the desk. In this case, you would really need to listen to the output of the desk via headphones. Also if you are broadcasting from a noisy room, monitoring the output through a good pair of headphones is a great idea.

A quick word on headphones...there are two basic types - "closed cup", and "open cup"...a closed cup headphone is one that blocks out noise from the outside world as much as possible.



You can see that the 'cup' (the bit that goes around your ear) is 'closed' - it is trying to prevent sound seeping into your ear so that you just hear the audio that is being generated by the headphone speakers.

An 'open cup' looks like this:



In these type of headphones the cups are open, sometimes looking like a light gauze, and these are intended for quiet environments. Both types are as good as each other but the design philosophy designates whether they are best to use in quiet or

noisy environments. I recommend you go for the closed cup unless you have a very quiet studio, in which case you can choose either.

### **HEADPHONE VOLUME**

The knob to control your headphone volume. Note : changing the headphone volume has no effect on the main volume - you can twiddle the headphone all you like and it won't effect the level of volume coming from the main outputs of the desk.

### **STEREO INPUTS**

The place where you plug in your stereo audio devices...usually this will be in the form of a Jack plug. On channel 1 and 2 you will also see some inputs for a Jack, these are the mono input sockets.

### **MIC INPUTS**

Where you plug in a mic which uses a XLR connector. Also, note that some small microphones do not have XLR connectors. In this case you would plug them into the mono input socket just below where the mic input is. Microphones are generally (but not always) mono.

ok...now we move on to some things you might not be familiar with, PAN, GAIN, and EQ. these are core components of the mixing desk and mastering these will make your life a lot easier and the output closer to what you want to achieve.

### **GAIN CONTROL**

Gain control is pretty much another volume control. Gain amplifies the signal on that channel and is usually used to boost microphone inputs incase the incoming signal is too low. Be careful when using Gain - a little is better than more...this is because when you turn up the Gain you also amplify any noise that might be on that channel. Sometimes noise is created in a desk because its not a very well made mixing desk, or it might be that your microphone isn't too good, or it might be that the cables are crappy...whatever it is, you probably don't want it, however turning up the gain will amplify both the signal that you want (from the audio source) and the noise....so keep an ear on this...

### **PAN**

Panning is just as it sounds...moving the sound from left to right...it could also be called 'balance' which is something most people will be familiar with. Experiment with this with headphones on and you will clearly understand what I mean.

### **EQ**

Alright! the science! So...you might already understand that any sound is made up of assorted frequencies. These frequencies could be classified on the most simplest level as 'high' and 'low'...a high frequency sound, is something that has a high pitch, a whistle for example. A sound made up of low frequencies is something that sounds bassy, like ..well...like a bass! When you hear someone playing a bass, they are making very low frequency sounds.

Ok...so most sounds are 'pure frequencies' they are comprised of collections of frequencies, some high and some low.

If we look at the desk we are using as an example, we see that there are three 'EQ' knobs for each channel. These control the relative loudness of 'high', mid' and 'low' frequencies for each channel. You can infact break down the control of the frequencies into much smaller groups to give much more control, but in most simple desks you get only a few groups.

The desk above has three groups, as I said - 'high', 'mid', and 'low'. Playing a sound through a channel and playing with the EQ will effect the sound quite dramatically...try it out and see what happens...'EQ' by the way, is short for 'equalisation'...this might be a phrase you have heard before. Tweaking the EQ helps sculpt the sound you want. Its good practice whenever you setup the desk, to set every EQ knob to the middle (or 'zero') and then go from there....this is also true for PAN and GAIN. As a general rule...don't go too heavy on the EQ, PAN or GAIN!!!!

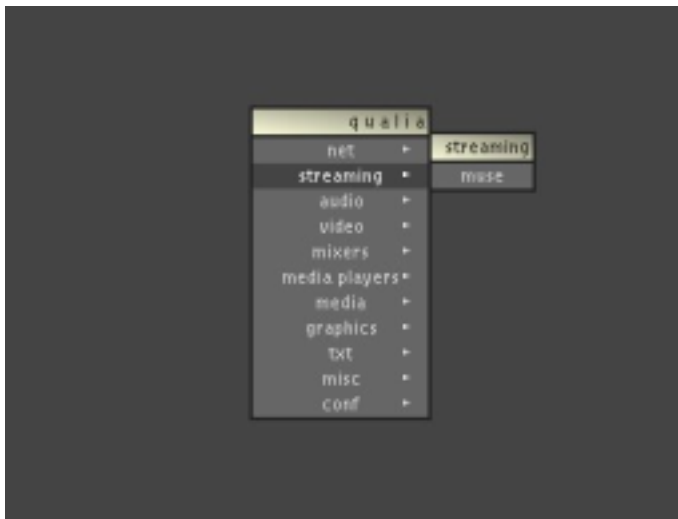


## INTRODUCTION TO LINUX WINDOWING SYSTEMS

The main difference you will first notice between Linux and whatever operating system you are used to using is the desktop environment. The desktop environment, in case you are unsure of what I mean, is the graphic interface that you use to do things on your computer, this is also known as a 'window' environment (don't confuse the use of this term with 'Microsoft Windows'). For example, I use a windowing environment on my Linux Laptop called 'fluxbox' and when I log into my computer I see this:



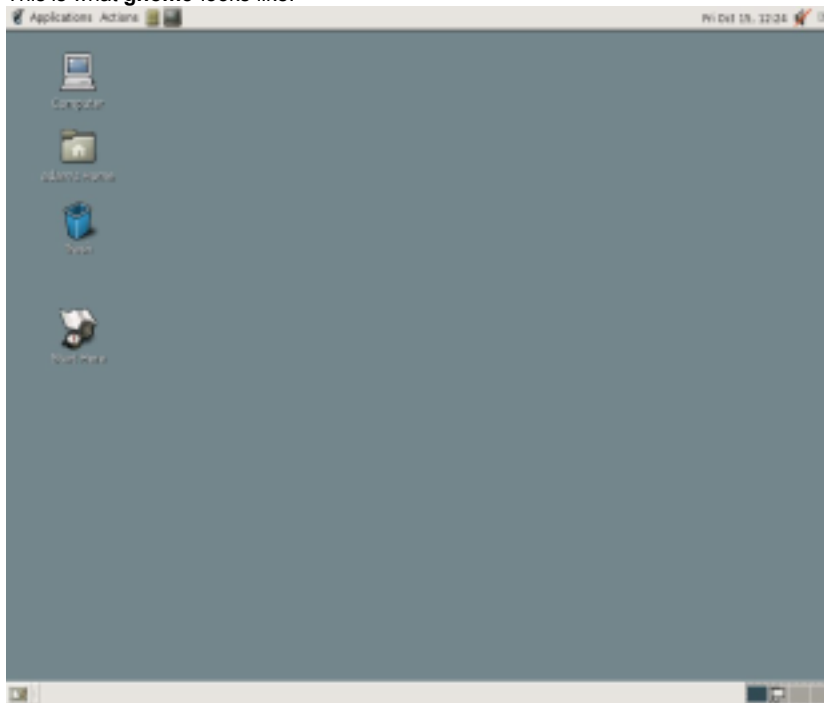
Now...you might be thinking 'oh my goodness your machine is not working! its all grey!'...well, yep it is working and yes it is meant to look like this. This is because **fluxbox** uses quite a different set of graphical navigation concepts than what you might be used to with Macintosh or Microsoft Windows. To access my files I don't use the 'start' menu like you would do in Microsoft Windows, or the quick start bar that OSX has...instead I right click on the window and I get a list of the programs that I frequently use, like so:



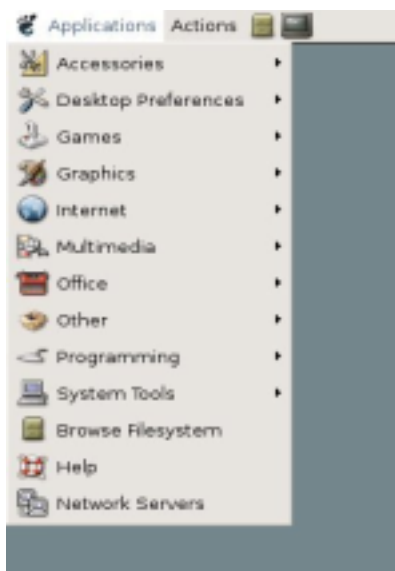
If you have a blank screen come up in your version of Linux then try a right click as I illustrated above. If a menu opens up (mine has been customised and is not intended to be a generic example) then you can slide along the categories until you get the application you need and click on that name. If you can't find the application name that you need then use this method to open a terminal and open the application by typing its name in the terminal.

Now, you probably won't be confronted with something as harsh as what my desktop currently looks like, you will most probably be presented with one of the two most popular Linux windowing environments - **gnome** and **KDE**

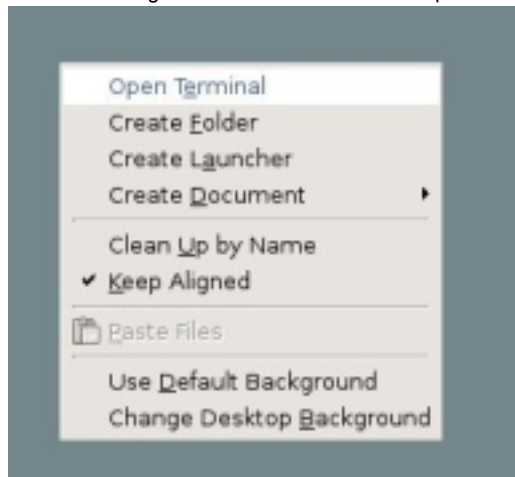
This is what **gnome** looks like:



This might look a little more friendly to you. It should be said that **gnome**, like many Linux windowing environments, can be configured many ways (a lot more than OSX or Microsoft Windows) so chances are even if you are running **gnome** it won't look exactly like what you see above. But...general principles are, that **gnome** has a bar through which you can access applications (similar to how Microsoft Windows works) but this bar is usually at the top of the screen. By clicking on the 'Applications' menu you can access most of the applications installed on the computer...

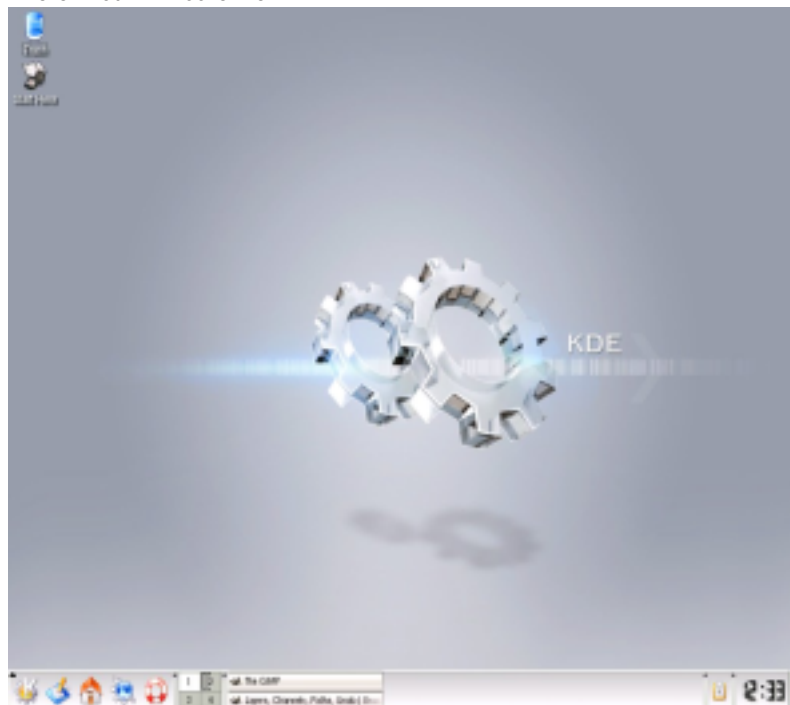


You can also right click and access another 'quick start' menu...like so:

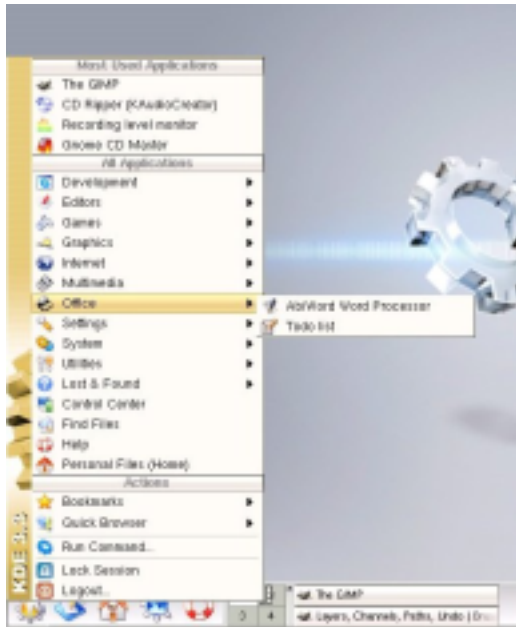


So if you get really stuck you can access the terminal through this menu and then browse the file system and open applications by typing their name directly in the terminal.

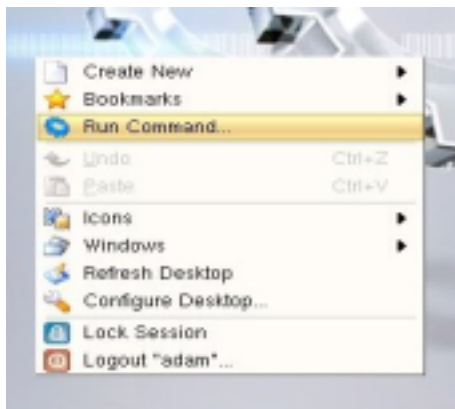
This is what **KDE** looks like:



This looks very much like Microsoft Windows. Remember again that **KDE** can be configured many many different ways so what you have on your computer might not look like this. But ...if it looks something like this then you can access applications from the 'K' menu at the bottom left of the screen:



And lastly if you get stuck you can right click and instead of opening the terminal you have an option to use the 'run' window:



This will then open a small application from which you can open other applications by typing the name of the application in the field provided.

## INTRODUCTION TO LINUX FILE STRUCTURE

If you open your terminal and type the following (followed by a return):

```
cd /
```

You will be placed in the top in the top directory of the Linux file system, if you then type the following:

```
ls -al
```

You will see something similar to this:

```
sh-2.05b$ ls -al
total 80
drwxr-xr-x 19 root root 4096 Oct  9 13:57 .
drwxr-xr-x 19 root root 4096 Oct  9 13:57 ..
drwxr-xr-x  2 root root 4096 Oct  5 09:31 bin
drwxr-xr-x  3 root root 4096 Oct  9 21:47 boot
drwxr-xr-x  1 root root    0 Jan  1  1970 dev
drwxr-xr-x 71 root root 4096 Oct 15 11:35 etc
drwxr-xr-x  4 root root 4096 Oct  9 19:21 home
drwxr-xr-x  8 root root 4096 Sep 18 23:29 lib
drwx----- 2 root root 16384 Sep 18 20:06 lost+found
drwxr-xr-x  9 root root 4096 Oct  9 16:36 mnt
drwxr-xr-x  9 root root 4096 Oct  8 23:20 opt
dr-xr-xr-x 64 root root    0 Oct 15 11:35 proc
drwx----- 75 root root 8192 Oct 15 12:35 root
drwxr-xr-x  2 root root 4096 Sep 23 18:58 sbin
drwxr-xr-x  9 root root    0 Oct 15 11:35 sys
drwxrwxrwt 60 root root 4096 Oct 15 12:36 tmp
drwxr-xr-x 17 root root 4096 Oct  5 09:31 usr
drwxr-xr-x 15 root root 4096 Oct  9 13:57 var
```

The above commands (**cd** and **ls -al**) are explained in more detail later in the manual, so I won't go into them now, all we are interested in is the output from these commands. The above listing is pretty much the standard directory structure for Linux. Each name on the far right represents a directory and each directory contains files and directories that are specific to that directory. the **lib** directory, for example contains code libraries that the software on your system use. For now you only need to be concerned with one directory, and that is the **home** directory, this directory contains folders that have names corresponding to each user of the machine. If you log in as **'adam'** for example (assuming this user exists on the system which it probably doesn't) then you will be logged into a directory in the **home** directory with your username (ie. 'adam' in this examples) as its name.

We will go more into this later in the manual. So don't worry too much about this now. The important thing to know is that there is this directory structure. The other important thing to know is that Linux is mostly made up of text files, so you can change almost every facet of Linux - how it looks and works - by just editing appropriate text files. In Microsoft Windows and Macintosh environments you usually do these kind of changes using small applications with a graphic user interface (GUI). In Microsoft Windows, for example, if you want to change the resolution of your display you use the 'display' control panel located in the 'control panels' directory. In Linux you can do this by editing a text file.

This has some advantages - it gives you a lot more control for example. But it also has some disadvantages - it can be hard to learn which files to edit and what to change (for example). Sometimes, to ease the transition to Linux from other operating systems, you will find there are configuration softwares for Linux installed on your system which use GUIs similar to Macintosh and Microsoft Windows. This is not always the case however, and so eventually you find yourself doing this manually with a text editor.



## SOME LINUX COMMANDS YOU SHOULD KNOW

There are some basic Linux commands that will make your life in Linux easier....

Here are the commands I think you should be fluent with:

```
man
ls
cd
mkdir
mv
rm
locate / slocate
ping
cp
pwd
tab
```

And some others that would be good to know:

```
ldconfig
updatedb
./configure
make
make install
tar
more
whereis
```

So, lets have a look at each. It is a good idea to experiment with these commands, there isn't much damage that you can do, and if there is a possibility one of the commands could accidentally create havoc (by someone thats not used to using it) then I will make a note in the comments below. In general though, just try each of these out in a terminal:



### man

This is a good command to start with because this accesses the help pages for the entire system. **man** is short for 'manual' and if you type this command followed by a space, and then the name of another command then you will get a help page displayed in the terminal telling you about the command. For example, typing :

```
man ls
```

Well give you a terminal window filled with information about the **ls** command. The format of this helppage might be a bit confusing, so just have a browse and don't get too confused. The part you need to be interested in most is the description of the command (ie. what it does). To scroll down the manual page press **space** to quit the man page press **q**

Try some man commands and read about the other commands I have listed above. There is also another help system that works the same way but instead of typing **man** and the command, you type **info** and the command like so:

```
info ls
```

Experiment!

### **ls**

the **ls** command is the 'list' command and this will list the contents of any directory you are in. Try typing this command in a terminal window and see what you get. Now, one feature of Linux commands is that you can add various parameters to them. This is quite a simple thing to do and can refine how you use the command. Usually these parameters are added to the command by typing a '-' directly after the command and then the parameter names or abbreviations. For example if I type the following:

```
ls -l
```

Then I am passing the 'l' parameter to the **ls** command. The 'l' parameter is more or less short for 'list' and refers to a type of format that the **ls** information should be displayed in. This format gives more information than just typing the **ls** command by itself...try this out and see the difference. You might well ask 'how do I know what the parameters are for each command?'...this information can be found out in the **man** pages for each command and accessing these is easy (see above).

For the **ls** command I suggest you get familiar with the formats using **ls** by itself, as well as **ls -al** , and **ls -l**

### **cd**

The **cd** command is important as it is the command you will use most commonly to navigate through the filesystem on your machine. **cd** means **C**hange **D**irectory, or more accurately **C**hange **W**orking **D**irectroy. Try this command out typing **ls** so you get a listing of all the files and folders in the directory you are currently in. Then try typing out one of the file or directory names you see after the **cd** command, for example if there was a file called 'me.txt' I could type out the following:

```
cd me.txt
```

This will give an error because you can't change to a directory if it is a file. Its good to try this so that you understand that you can't do any damage by making a mistake with **cd** . Now to try changing to a directory, I don't know about your file system so I am imaging there is a directory called 'src'. If I wanted to change to this directory I would type:

```
cd src
```

If I am succesful then the terminal won't throw up an error, if I fail it will tell me so. If you fail it will be because either you don't have permissions to enter the directory you want to enter, or the directory simply doesn't exist.

Now...a word about the Linux filesystem. Generally, if the system has been set up nicely for you, you will be working in your home directory. This is generally located in a set place in Linux. To find your home directory first type the following:

```
cd /
```

This will pace you in the topmost directory on your computers filesystem. So then if you type **ls** you will see something like this:



```
xtenn
bash-2.05b# ls
bin  data  etc  lib      mt  proc  sbin  tap  var
boot dev  home lost+found  opt  root  sys  usr
bash-2.05b#
```

This shows the list of directories on your machine at the topmost level of the file structure. There are some important directories here, but you need to be most concerned with the one name 'home'. To change to this directory type :

```
cd home
```

Now if you type **ls** you will see some more directories, and hopefully one has your username in it, this is your home directory. Now, we have been navigating to this using relative positioning, that is - if I am in the topmost directory and I type **cd home** then I will be placed in the 'home' directory where all the users individual home directories are kept. I can type **cd home** because the 'home' directory is in the directory I am currently in. If I was somewhere else on the file system and I typed **cd home** I would get an error (you can see this by typing **cd home** from the 'home' directory itself). If you need to however you can use *absolute paths* to the directory you wish to get to. As an example if I was in some dark corner of my filesystem and I need to get quickly to the 'home' directory I would type:

```
cd /home
```

If I needed to get to a directory under the home directory (lets say I have a directory in there called "adam") I would type:

```
cd /home/adam
```

### **mkdir**

This is the command you used to create a directory and is short for **Make Directory**. To use this simply type the name of the directory you want to create after the **mkdir** command as so:

```
mkdir bleep
```

The above command will create a directory *in the current directory I am in* called 'bleep'. If there already was a directory with this name, then it won't over-write the existing directory and I will get an error. Try creating some directories.

### **pwd**

If you get lost and you don't know where you are in the filesystem you can always type **pwd** and it will tell you. This command gives you the location path or absolute path to where you are, for example if I am in my 'adam' home directory the output of the **pwd** command will be:  
/home/adam

Experiment with changing directories with **cd** and then typing **pwd** to see where you are.

## **mv**

This command is short for **move**. It is as it sounds, **mv** allows you to move files around on the filesystem, if we have to go into Mac and Windows analogies, then this command is like *cut and paste* all rolled into one. To use **mv** you must first type the command, followed by the file you want to move (in absolute paths or relative paths *including the filename*) and then the place where you want to move the file to (in absolute or relative paths). For example if I wanted to move a file 'me.txt' from my current directory to the /usr/bin directory I would type the following:

```
mv me.txt /usr/bin
```

Note: I don't have to type the filename in the path name where I want to move the file *unless I also wish to change the name of the file*. If for example while I was moving 'me.txt' I wanted to change the filename to 'you.txt' I would type:

```
mv me.txt /usr/bin/you.txt
```

If I just wanted to rename the file and not move it I could use the **mv** like a rename command by typing this:

```
mv me.txt you.txt
```

getit?!! Note that when you use **mv** you are moving the file *not* copying it, so the original will be moved and won't exist in the same place you moved it from. Now, also be a bit careful because you can overwrite files accidentally, if for example I moved one file to a directory with a file of the same name, then the file I am moving will overwrite that file. Then you could be in trouble...so just be a wee bit careful when using **mv**

## **rm**

On the other hand, here is a command you should be very wary about using. **rm** is short for **rem**ove and is the command you use if you wish to delete a file or directory (and its contents). To use this command type **rm** followed by the name of the file you wish to destroy for good. To remove a directory you can use the same command with the parameter **-R** like so:

```
rm -R directoryname
```

Where 'directoryname' is of course the name of the directory you wish to remove. You can also use **rmdir** for this which (you guessed it) is short for **rem**ove **dir**ectory. Be EXTREMELY careful when using **rm** or **rmdir** ... if used unwisely it could be the end of your operating system.

## **locate / slocate**

These commands help you find files on your filesystem. A reference to all files on your system are stored in a database which is updated periodically by using the **updatedb** command. To find a file simply type either **locate** or **slocate** (depending which, if any, is installed on your computer) followed by part of the name of the file or directory you are looking for. For example if I am looking for the file 'icecast.conf' I would type:

```
slocate icecast.conf
```

And this will tell me exactly where the 'icecast.conf' file is...if I don't get any reply from typing this it means that either the file doesn't exist on my system or it exists but my database doesn't know where it is. In this later scenario I would type **updatedb** and try again.

**locate** and **slocate** are nice commands, you can't destroy anything by using them so experiment as much as you like. **updatedb** might take a while to run sometimes if you haven't run the command recently or if you have a slow machine, it can also use a lot of CPU power on slow machines so never use it while you are doing something else 'mission critical' on your machine like streaming.

You might also like to experiment with **whereis** and **find** to look for files on your system.

## **cp**

This is short for ...guesses?...**copy**. Use it like **mv**, the only difference is that it leaves the original file where it was while also creating a copy.

## **ping**

**ping** is handy for streamers...not usually included in the top 10 commands you need to know but its handy if you need to know if you are online, which is pretty central to the idea of streaming (unless you are just testing, or streaming over a local network). **ping** sends a request to any computer on the net that you like to check to see if your computer can reach it. There is a lot to **ping** but we

want to use it just to check if you are online at all. In this case just type **ping** followed by a url that you know, for example it might be a good idea to try the following:

```
ping www.cnn.com
```

I usually use :

```
ping xs4all.nl
```

Because I used to work for xs4all and I know that only the end of time itself would take them offline ;)

If you are online you will get some information coming back through the terminal to tell you so...this will keep scrolling so to stop it type **ctrl** and **c** and it will halt the ping process. If you get no response from **ping** then you are probably offline and thats the first thing you will probably need to fix if you want to stream. There are a few things that you should know about **ping** ... the first is, some machines online don't answer ping requests for security and other reasons...so make sure you really know that the machine you are pinging does reply to **ping** requests before you are in that so called 'mission critical' situation. Also, some internet connections won't allow **ping** traffic...for example, while I am writing this I am in an internet cafe in Riga,...its a fast connection but I can't **ping** , this perhaps because they think only evil hackers use **ping** so they have some paranoid network security disallowing all sorts of useful things.... The other things is that if you know a friendly machine that replies to pings, you might like to leave a terminal open that pings that server while you are streaming...I say this tentatively because it comes close to breaching internet etiquette (netiquette), but if you know the person operating the server you could ask them first if this is ok, then if you keep the **ping** running while you are online so you can keep an eye on the connection...this is handy if you are unsure of the network environment and suspect it might time-out from time to time which is more or less a show stopper if you are trying to send a continuous stream. There are other ways to monitor connections, but this is perhaps one of the simplest.

#### **tab**

**tab** is not so much a command as a keystroke...every keyboard has a **tab** key (i think), and its a very useful thing to have in Linux. You might have used this keystroke before to indent words in a word processor. You can still do this in Linux word processors, but when you use **tab** in the Linux terminal it becomes such a time saver that when you master it you will be using it *all* the time. Essentially the **tab** keystroke is like an auto-complete. If, for example, I want to move the file 'dsjkdshdsdsjhd\_s\_dsjw22.txt' somewhere with the **mv** command I can either type out every letter of the stupid filename, or I can type **mv** (for 'move') followed by the first few letters of the filename and press **tab**. The rest of the filename will be automagically filled in. If the filename is not filled in it means that there are several files (or directories) that start with those first few letters I typed. To remedy this I could type a few more letters of the filename and press **tab** again, or to help me out I could press **tab** twice and it will give me a list of files that start with those letters.

**tab** is your friend, use it a lot.

#### **OTHER COMMANDS**

At the beginning of this section I said there were a few 'other' commands that might also be good to know, they were:

```
ldconfig
updatedb
./configure
make
make install
tar
more
whereis
```

I have already talked about some of them, namely **whereis** and **updatedb**. The others might be useful if you are installing software, so I will talk about how to install something with Linux, but before that a little word about the **more** command.

**more** is used if you want to control the overly verbose output of any command to the terminal. If for example, I am in a directory which contains 1000 files and I type **ls** the output of the command won't fit nicely into my little terminal window so it will go scrolling past faster than is useful. To slow it down so I can read the output we follow the command with **| more** like so:

```
ls | more
```

If I used this in my 1000 file directory I get one page at a time of output and pressing the **spacebar** shows the next page. Pressing **q** quits more. Ok, so you might be wondering what the funny straight line is in the above command...well, this is known as the **pipe** command, and you can't get past go without it. Essentially **pipe** allows you to combine commands together to control the kind of output

you get, usually its used to refine a command (which is what the command parameters also do). So, when you get really fluent with these commands you can write things that look more like equations but are really efficient ways of using standard commands...**pipe** will be central to enhancing your efficiency.

## CONNECTING TO THE INTERNET WITH LINUX

If you start your Linux computer and perhaps you have opened a browser (try typing **mozilla** in a terminal) but you can't get online then you might have a problem. The easiest way to resolve this is to ask the nearest Linux guru you know to help you out because there are many things that could be causing the problem. However, before asking the Linux guru you might want to try this command:

```
dhcpcd eth0
```

This command tries to make a connection with the internet using your ethernet card. The command requests an **IP** address (a number unique to your computer that helps you access the internet) from the local computer that takes care of handing out **IP** addresses. Trying this command has one of three possible outcomes:

1. it hangs the terminal for a long time without giving a response
2. it returns an error saying the command isn't known
3. you are successfully given an IP address (unfortunately you won't be given any feedback to say this so you just have to try your browser again to check).

In the case of (1) above...it means that for some reason you are not going to be given an **IP** address, so you should contact your system administrator and ask for help. The most common reason for this error is that you need a **dedicated IP** address...but don't worry about what this means, but it is something you may wish to research another day.

In the case of (2) above, it means you don't have the command **dhcpcd** installed or you don't have priviledges to use that command. In this case you should talk to the Linux guru or your system administrator.

In the case of (3) above, you are very lucky.

If you get *really* ambitious, try learning about the command **ifconfig**. This is also important to Linux networking but you may wish until you have learnt a little more about Linux before reading about this. There is a handy help file on this command, which you can access by typing:

```
man ifconfig
```

in the terminal.

## INSTALLING SOFTWARE ON LINUX

Well, installing software on Linux is a broad subject because each version of Linux has its own **package management system**. For example, if you wanted to install **Abiword** (a really great word processor - this manual is written using Abiword) on a Gentoo Linux system then you would simply type the following in a terminal:

```
emerge abiword
```

Then, if you are online, the **emerge** command downloads all the appropriate sources and installs **Abiword** for you. Easy! Debian also has the **apt-get** command and you would use it as so:

```
apt-get install abiword
```

There are many many package management systems...too many to describe here so I will go into a little detail on how to do what is called a **source installation**. To do this you need to first download an application from a website somewhere. The source packages are usually appended by a **.tag.gz** suffix, which is the suffix for a type of compression favoured by those creating source packages.

Installing from source works on any Linux system, so its a good process to know, and it more or less follows this route once you have a source package:

```
tar zxvf packagename.tar.gz
```

Where 'packagename' in the example above is the actual name of your package that you wish to install. The **tar** command followed by the parameters **zxvf** uncompresses a **tar.gz** file and creates a new directory with all the extracted sources. Now you must change your working directory to this new directory using the **cd** command (see above). usually the new directory name is the name of the compressed source package minus the **.tar.gz** suffix. For example, if my package really was called 'packagename.tar.gz' then after running the **tar zxvf** command on it I would be left with a new directory called 'packagename' and then I would type **cd packagename** to enter this new directory. If you are not sure of the name of the newly created package type **ls**

Alright...so then, once inside the new directory, we want to start the actual installation process. To do this 99% of the time you will need to type the following:

```
./configure
```

Ok, so this isn't really a command. Each installation package usually has a script called 'configure'. By putting a dot and then a slash before the name of the script (ie. 'configure') you are telling Linux to execute (run) that script. The configure script then does its stuff, checking what kind of machine you have, what you already have installed, what kind of Linux you are running etc etc etc.

The most common problem that will occur at this stage is that the configure script will halt and tell you that software library that the new software depends on is missing. This can be a pain in the ass which is why people invented package management systems. However if you do experience this error then you need to use a search engine to find out what software the error message is talking about and where to get it, then start the installation process again with this new package. I am not kidding when I say that this can sometimes mean an installation can take *days* while you search and download all the packages you need. But I don't think this will be the case with the software covered by the scope of this manual (fingers crossed).

So, lets assume you don't get any errors created by running the configure script...in which case you are lucky and you should thank whatever angel is looking over you ;) Now...the next command to type in the install process is **make** like so:

```
make
```

This command actually makes the software for you according to the information that your configure script has gathered about your machine. You will then end up with a whole lot of compiled files which in total makes up your software. The **make** process can take a while depending on the speed of your machine and the size of the package sources you are installing and also of course if you are running other applications this will slow down the process. However you are not finished yet. Now type the following:

```
make install
```

this will install the newly created software in the correct places in your system. So now you just need to type the name of the application in your terminal window and it should run. If it doesnt run and throws an error, a common remedy is to type **ldconfi** and then try again. **ldconfi** updates the system so that your operating system knows there are new library files etc.

phew....thats all the commands you need to know for now :)

## INTRODUCTION TO LINUX SOUND ARCHITECTURE

Well, you don't need to know too much about the sound architecture itself...here is some information that outlines the original sound system for Linux (OSS) and the currently used Advanced Linux Sound Architecture (ALSA).

### **OSS - Open Sound System and ALSA - Advanced Linux Sound Architecture**

Once upon a time the many and various UNIX architectures all had their own ways for dealing with sound. So porting an audio application from one to the other required also converting code to handle the sound on the new operating system. OSS (formerly known as VoxWare, USS, and T ASD) was an attempt to remedy this situation and the people at 4Front Technologies (<http://www.opensound.com/> and <http://www.4front-tech.com/>) developed source code that could be compiled and ran across all these UNIX variants including Linux, SCO OpenUNIX, Solaris, IBM AIX, FreeBSD, OpenBSD, and NetBSD

OSS is hence the original sound system architecture used by the Linux kernel. However tOSS IS only used until kernel version 2.4.x. Since kernel version 2.5 the Advanced Linux Sound Architecture (ALSA) has been the default sound management system.

ALSA supports OSS applications by running emulation (represented by the modules `snd-pcm-oss`, `snd-mixer-oss` and `snd-seq-oss`). To load this emulation automagically you will need to hand edit the `/etc/modules.conf` file to include these lines:

```
alias sound-service-0-0 snd-mixer-oss
alias sound-service-0-1 snd-seq-oss
alias sound-service-0-3 snd-pcm-oss
alias sound-service-0-8 snd-seq-oss
alias sound-service-0-12 snd-pcm-oss
```

OSS-midi is emulated directly in the ALSA code base and doesn't require a module for emulation.

If you would like to see the current status of OSS in your kernel then look here `/proc/asound/oss/sndstat`/`proc/asound/oss/sndstat` by typing in a terminal:

```
cat /proc/asound/oss/sndstat
```

For more information on OSS emultaion by ALSA look here:  
<http://www.alsa-project.org/~iwai/OSS-Emulation.html>

To check if your soundcard is supported by OSS look here:  
<http://www.4front-tech.com/osshw.html>

OSS make a free version of their sound system and a commercial variety.

Since kernel 2.5 ALSA is the default sound subsystem in Linux. ALSA is open source (GPL).

## LINUX SOUND SOFTWARE YOU NEED TO KNOW - MIXERS

Alright...there are some softwares for operating sound on Linux that you should be aware of. First of all it should be said that Linux is an operating system and not a Windowing system. Windowing systems are the things we are most familiar with when we use our computers ...it could be described as the graphical user interface that we use to access our files and do all the things we do with our computers. Mac and Microsoft Windows have their own windowing systems, and you will be familiar with those. Linux however, has many windowing systems that have been made by various groups of programmers....the most common are KDE and GNOME. KDE largely follows Microsoft Windows design concepts, while GNOME doesn't follow them so closely...The look and feel is quite different with these systems...I use another system which is radically different called 'Windowmaker'....

I raise this point because its tricky to explain a Linux application when you might be using a computer with one of a dozen windowing environments that run on top of Linux. So, I will go to the basics and assume that somehow you can get a terminal window open on Linux...how you do that is up to each Windowing environment, but once you have discoverde how to open a terminal then we should continue...a terminal looks something like this:



This is known as an xterm...there are very many varieties of the terminal...don't worry about it too much, just get one open! ;)

The cool thing about a terminal is that it gives you access to the entire operating system simply by typing a few commands...this is the 'command line' interface you might have heard of and is how all the computers in the old days used to work...the command line interface gives a lot of control to you the user, which means you can be very efficient but it also means you have to be careful otherwise you could do a lot of unintentional damage to your file system, infact its very easy to screw up the system completely...this is why Linux has a user system, in this system the 'root' user is the omni-present 'god-like' user on the machine. There is only one root user for each Linux box and when logged in as root you can do anything. It is not a good idead therefore to log in as root. Logging in as another user with a lesser domain of influence is the best bet, to save unintentional blunders.

So,...if someone else has set up the Linux machine you are about to use, ask them to create you an account and tell them its for streaming so you will need access to all things regarding audio...hopefully they will know what you mean and know how to do it...if the root user is you and you dont know how to do this (maybe you have just set up your first Linux box) then look online for the Linux 'useradd' and 'usermod' commands...also you can look at the internal Linux helpfiles anytime by typing 'man' (short for 'manual') followed by the command you want to find out about. For example you might type:

```
man usermod
```

ok...so leaving that behind for now, we will imagine that the computer you are about to try streaming from has been set up correctly and you have logged in from a user other than 'root', and you have a terminal open similar to the one above...well done!

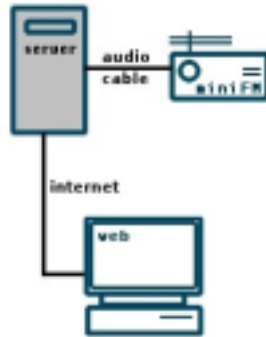
Now I am hoping that you have a software mixer of some variety installed in your machine. There are many varieties of mixer...lets look at some of the most common....alsamixer, rexima, kmix, aumix

Try typing each of the application names listed above in the terminal window followed by a 'return' or 'enter' keystroke...if I type :

```
aumix
```







...thanks again to Irrational.org for their cool icons...

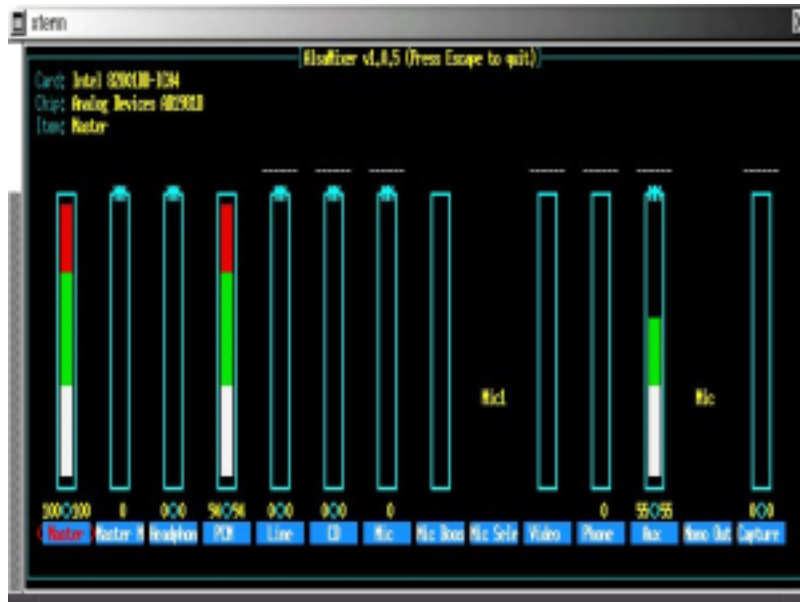
That meant that everything coming through the soundcard of the Linux machine would be broadcast live on the radio station (FM). The Linux machine was also connected to the internet so we could log in remotely. If we had something playing on the Linux machine and it was playing too loud, we could just log in with a terminal and turn it down using **rexima**. This comes particularly important if you are running a set up such as I have just described and the internet connection is slow, or either the remote machine, or the machine you are using to log in, are slow....'curses' interface applications are perfect for these circumstances because they require minimal bandwidth or CPU resources.

**rexima** works the same as **aumix** except that you use the keyboard instead on the mouse....try using the arrow keys to navigate through **rexima** and alter the volumes.

alright...had fun? now lets try **alsamixer**...type **alsamixer** in the terminal after quitting **rexima** (to do this just press 'q') ...and hit return, and you should see this:



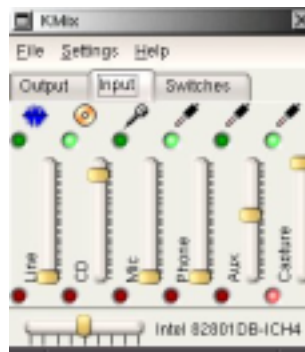
If you grab the bottom right of the terminal with your mouse and click and drag, you can stretch the **alsamixer** wider:



This is also a terminal application and is written with the 'ncurses' library which is similar to the 'curses' library used by **roxima**.

The navigation principles are the same. To quit however you have to press 'ctrl' and 'c' simultaneously.

Lastly lets look at **kmix**...this is my least favorite mixer as it belongs to the KDE family of software which I personally think looks ugly. So type **kmix** in the terminal and you should get something like this:



Again, the principle is the same, they just look different.

## LINUX SOUND SOFTWARE YOU NEED TO KNOW - XMMS

Ok...now to the players...you need to know how to use a simple media player under Linux because they are useful for checking whether your stream is actually working or not. In other words, once you have a stream running it's a good idea to connect to the streaming server with a media player to check:

1. the stream is there
2. the audio sounds alright

I will just say that **xmms** is a good choice for doing this because it supports both mp3 and ogg for playback, it handles Icecast2 streams nicely (the server we will discuss later), and if you come from a Mac or Windows background the interface is very familiar, and lastly **xmms** comes installed default in most Linux systems. You should just be able to type:

```
xmms
```

in a terminal and it will open the player. If it's not installed look at the section on how to install software and download **xmms** from <http://www.xmms.org> and install it (good luck!).

If however it is installed then you can have a play with it and see what it does. Basically **xmms** is modeled on the Microsoft Windows player software **winamp** and hence the interface is almost exactly the same. It is for this reason a lot of Linux junkies don't like **xmms** but in my opinion it's the best player for Linux available. Why? Because it has a lot of sundry developments that have been done by third parties that occur because (unlike **mplayer** another popular player) it has a very nice plug-in structure. This means coders can extend the functionality of **xmms** quite easily. This has in fact happened and if you have **xmms** installed you can now look at the plugin pages on the **xmms** site (<http://www.xmms.org>) and see what kind of functionality you can get from **xmms**. You can, for example, plugin the **Liveline** plugin for **xmms** or the **Oddcast** plugin and use **xmms** itself to send streams.

Anyway, we won't use it for this, we wish to use it just to monitor streams. So let's have a quick peak at the interface of **xmms**



The interface elements you need to worry about are but a few:



To open a stream that is coming live from a streaming server you need to right click on the player and choose 'Open Location', or alternatively press **ctrl** and **I** simultaneously. From this point you need to enter the URL (location) of the stream (more on this later) and then click 'ok', the stream should start playing providing you are online and the stream actually exists!

## LINUX SOUND SOFTWARE YOU NEED TO KNOW - TEXT EDITORS

Righto...so you will be thinking 'text editors! i thought this was about streaming media!' ...well, if you don't know how to use a text editor in Linux then you can't really get too far. Reading 'README' files and 'INSTALL' files will be a necessity quite early on when learning Linux. You will also need a text editor to need to change webpages to link to your live stream or to make playlist or m3u files to link to the live stream (more on these later).

text editors is a topic that many books have been written about. So, how do we cover it here and do it justice...well its tricky. We can at best get a superficial glimpse. There isn't even enough time to write an intro to all the major text editors, so we will arbitrarily choose a couple: **nano**, **vim**, and **Abiword**. We will also look at **less** which is not an editor but is a command that allows you to read files on your system.

Lets start with **less**

This is a command that opens text files for read only. If, for example, the directory you are currently working in has a file called 'README', then try this command:

```
less README
```

and you should see this in the terminal:



To scroll use the up and down arrows, and to quit just type **q**

Obviously **less** is the easiest way to read files but unfortunately its not always installed on Linux systems, so try **less** first and if it fails try on of the others above. Remember that **less** will only allow you to read files. To edit files you will need a text editor or word processor (Sometimes there isn't much difference between the two). **nano** and **vim** are text editors which are commonly used by programmers for working on Code, and **Abiword** is a fully fledged word processor used for office-like applications.

So, first check if these are installed on your system by typing each name. I would say you will probably have **vim** installed, and might have **Abiword** installed and probably don't have **nano** installed (which is a shame because it is a very simple text editor and does all we need to do within the scope of this manual). If you don't have any of these installed you can try installing them by downloading the source packages and following the install process described earlier, or you can try installing with the package management system that your version of Linux uses. If you don't wish to install anything new then maybe try typing the names of some of the other text editors available, but if these are installed you will have to go online and have to research how to use them yourself. Try these in the terminal:

```
vi
gvim
emacs
pico
soffice
fte
```

**vft**  
**gedit**  
**glimmer**

From the above list the easiest to use if you are familiar with word processors are **soffice** also known as 'Open Office', and **gedit**. Both of these work quite like **Abiword**, so if you don't have **Abiword** installed then try these two and follow have a look at the **Abiword** section below.

So...lets have a quick look at **vim**, assuming it is installed when you type **vim** in the terminal you will see something like this:



```
vim - Vi Improved
      version 6.3.25
      by Bram Moolenaar et al.
      Vim is open source and freely distributable

      Help your children in Uganda!
      type :help toq[Enter]> for information
      type :q[Enter]> to exit
      type :help[Enter]> or :?[] for on-line help
      type :help version[Enter]> for version info


      1/0-1  [0]
```

Incidentally if you have **vi** installed you will see pretty much the same thing.

Alright, well to open a file with **vim** it is best if you type the name of the file you wish to open after the **vim** command, so that **vim** opens with the file already loaded. For example if we wanted to read a README file in the same directory we are currently working in then just type:

```
vim README
```

This will open **vim** with the README file loaded as so:



```
dyne.org autoproductions & the FreakNet Radialab
      proudly present:

      MUSE (OO)
      codename "DUTURNIX"

      this is RISTIX SOFTWARE. Jah Rastafari Livvity bless your freedom!

      :: the Multiple Streaming Engine ::
      https://muse.dyne.org

      This application is being developed in the hope to provide the Free
      Software community a user friendly tool for network audio streaming,
      making life easier for independent free speech radios wanting to
      stream via http on icecast servers.

      MUSE is an application for mixing, encoding, and network streaming of
      sound; it can transmit an audio signal by mixing together sound taken
      from files or also network, recursively mixing more MUSE streams.

      MUSE can simultaneously mix up to 6 encoded audio bitstreams (from
      1.1 Top
```

Now to scroll up and down the file use the up and down arrows on your keyboard. To quit **vim** then type:

```
:q
```

There is really a lot to **vi** or **vim**, and I don't want to get into it here, but you should really know how to open a file (As above) and then edit a file. To edit a file in **vim** you need to first open the file, and then type :

i

Now, I am imagining **vim** is quite a bit different to any text editor you have used before, so perhaps some explanation is needed. **vim** open a file initially as a *read only* file. This means that when you first open the file with **vim** you are not allowed to change the file. **vim** has then a whole world of commands you can use to work on the file and each command is in the format:

: *command*

Where 'command' is the name of the command you wish to use. The commands are all designated by shortcuts. An 'i' , for example, is short for 'insert'. The following is a table of **vim** commands you should know:

command	action
i	insert text
:w	write changes to file
u (only used in read-only mode)	undo changes
:q	quit vim

In addition, by pressing the 'esc' (escape) key, you will tell **vim** to return to the original mode (read only). You must actually press escape before you execute any of the commands in **vim**. For example if I wanted to open the file 'README' and then alter some text, I would do the following, starting with typing:

```
vim README
```

in the terminal. This will open the 'README' file as explained above. Then if I wish to edit the file, I use my arrow keys to navigate to where I want to insert or delete some text. I then press:

i

This will put me in the insertion mode and now anything I type will appear in the document itself. When I have finished making the changes I will then press the 'esc' key, and finally to save the changes I press:

:w

This will write the file with the new changes. I then need to quit from **vim** so I press the escape key followed by :

:q

Now find a file and experiment. If you haven't used something like **vim** before then it might take some getting used to, so spend some time working out for yourself how **vim** works before you really need to use it.

now onto **nano**

**nano** is a really lightweight handy editor, and its a shame its not used more often. Try typing **nano** in the terminal followed by the filename of the file you want to open, in this case 'README', like so:

```
nano README
```

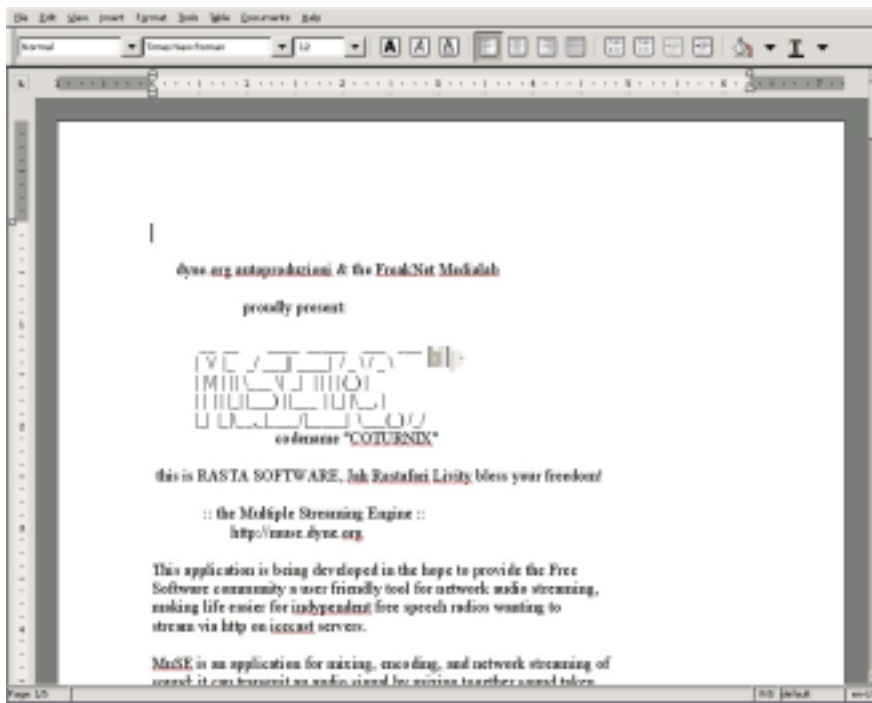


Again, the application opens in the terminal. Good if you need to edit material on a remote server. So to scroll you use the up and down arrows, and to quit you use **ctrl** and **x** pressed simultaneously. With **nano** if you need to edit a file then just start typing directly into the terminal window. If you want to save the changes press **ctrl** and **o** simultaneously and the file will be saved. Also if you quit **nano** and changes haven't been saved then **nano** politely asks you if you would like to save the changes, if you do press **y** else press **n**

Next up, try typing **abiword** in the terminal, before you press return try pressing the **tab** button incase **abiword** has a version suffix, for example my installed **abiword** is opened using the **abiword-2.0** command. So, you should then enter the filename after the command and this will open **abiword** with the README file preloaded, for example I would type:

```
abiword-2.0 README
```

and I would see this:



As you can see this looks like a word processor, and it is. To scroll up and down use the up and down arrows, and to quit use the quit function for the file menu.

If you have used a word processor before and you probably have, then you will be familiar with how **Abiword** works...all the commands are accessed through the top menu, and most of the commands you need (open,save,quit,undo etc) are accessed under the 'file' and 'edit' menus in the top tool bar.



## LINUX SOUND SOFTWARE YOU NEED TO KNOW - FTP

FTP stands for **file transfer protocol** and a software using this protocol for transferring files across the internet is called an **ftp application**. If you have done any web development then there is a good chance you have used a **ftp application**. These softwares allow you to transfer files to your webserver, and also to download those same files to your computer. If you are a web developer, for example, you probably have used an **ftp** application to download webpages from the site you are working on, then you have probably made some changes and used the same application to transfer the changed files back onto the webserver.

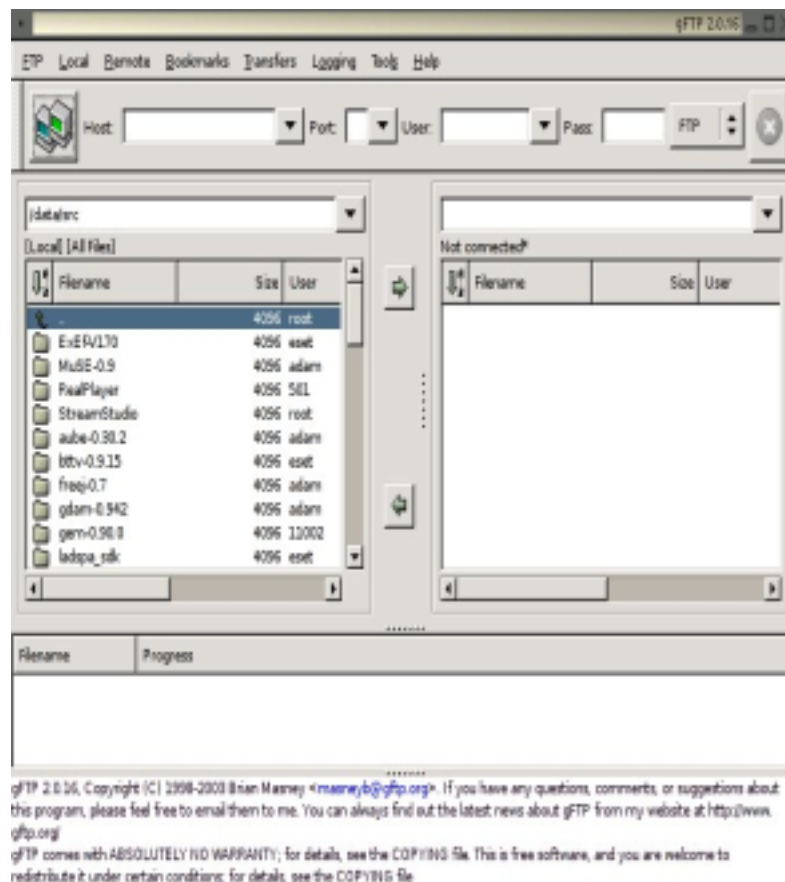
It is important to realise that when you are downloading these files you are copying the files ie. the original stays on the web server, however when you upload the files they overwrite any files with the same name...be careful with this as unthoughtful use of the **ftp** application can be disastrous.

There are quite a few of these softwares for Linux, the most common **gui** (graphical user interface) applications for **ftp** are **gftp**, and **NFTP**. There is also the **ftp** command which allows you to access **ftp** from the command line, we will look at this method and the **gftp** application. First up **gftp**. If you have **gftp** installed then you can simply type the following command:

```
gftp
```

and the application will open. If you don't have **gftp** installed then this command will give you an error, if this is the case then you need to either download the application ( download the sources from here - <http://gftp.seul.org/> or use your package management system to install) or use the **ftp** command, or alternatively use another Linux **ftp** application.

If you have **gftp** installed then you will have something like this on your desktop:



All **ftp applications** follow a similar concept but the layout is often quite different, however if you look at **gftp** then hopefully it will help you if you wish to use another application.

When you are given an account by an Internet Service Provider (ISP) then you will be given some 'account details' by the ISP. If you have asked for **ftp** access then those details will include the following:

**username**  
**password**  
**hostname**

The username and password combination is what you use so only you can access your webserver. Anyone trying to access your files will hopefully fail because they don't have the username and password necessary to gain access to your **ftp** account. However, you do have those details and so you can enter the information in the **gftp** fields labeled **user** and **pass** like so:



When you fill in the **pass** field with your password then you will notice that each character is replaced by an 'x', this is to prevent someone who maybe standing behind you seeing what your password is.

the next field 'host' is where you put your **hostname** information that the ISP gave you. The **hostname** usually looks like a URL, the type that you type in the location field in your browser, however sometimes it will be a series of numbers which is known as an **IP number**. You enter this information like so:



Then you press the computer icons on the right hand side of the top toolbar (see above) and **gftp** will attempt to connect to your webserver.

If you are successful then you will see a list of files appear in the right side of the application window, this is a list of the files that are located on your webserver. On the left side you will already have seen a file list, which is the list of files on the working directory of your computer. You swap files between the two by dragging and dropping.

If for some reason **gftp** did not connect to your webserver, then check you have your details correct for **password**, **username**, and **hostname**. Also, make sure the drop down box on the far right of the top tool bar (just to the right of the **pass** field, has **ftp** selected.

The chances are however, that you don't have **gftp** installed. So, if you are stuck for an application then hopefully you have the terminal **ftp** command installed. To test this simply type:

```
ftp
```

and you should see a prompt appear with 'ftp' following it (to exit this mode type 'exit' and you will be returned to your original terminal), otherwise if **ftp** isn't installed you will get an error. Lets hope its installed and follow a simple log-in process using the above details as an example. First type the following:

```
ftp www.radioqualia.net
```

In your case of course you would replace the hostname ('www.radioqualia.net') with your own hostname. In the case of the example above I would get this reply in my terminal:

```
sh-2.05b$ ftp www.radioqualia.net
Connected to radioqualia.va.com.au.
220 ProFTPD 1.2.7 Server (Virtual Artists FTP Service) [shiraz.va.com.au]
Name (www.radioqualia.net:eset):
```

Essentially what this is saying is that you have made a connection to the webserver and now the webserver has sent back a request asking you for to enter your username. In response type in your username and hit enter and you will get some feedback similar to this:

```
Name (www.radioqualia.net:eset): radioqualia
331 Password required for radioqualia.
Password:
```

Hence you should enter your password and you will get a response which looks something like this:

```
230 User radioqualia logged in.  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp>
```

If you see something like this then you have logged in successfully! Its important to remember that there are many types of ftp *server* softwares, and the output you receive in your terminal will depend on what kind of server you are negotiating a connection with. The above output is from ProFTPD but the output you receive could be quite different.

Once you have logged in you can now send commands to both your own computer *and* the remote machine (webserver). The type of commands available to you on the remote machine are very similar to the commands you already know how to use in the terminal. From the list of commands you already know you can use these in the terminal:

```
ls  
mkdir  
cd
```

Additionally the new commands you need to know are **remove**, **move**, **put**, **get**.

#### **remove**

This is the same as the Linux command **rm**. If you need to delete a file on the remote server then type:

```
remove file.txt
```

Where 'file.txt' should of course be changed for the name of the file that you wish to remove.

#### **move**

This is the same as the Linux **mv** command.

#### **cd**

Also the same as the Linux command.

#### **put**

This is a new command unlike any of the Linux commands you have learnt before. **put** allows you to transfer (put) files from your local computer onto the remote machine. For example, if I have the file 'index.html' on my computer (in the directory I am working in) then I would use the following command to put the file on the webserver I have just logged into:

```
put index.html
```

Note that if there is already a file on the server called 'index.html' then it will be overwritten by this new version so be careful, it often pays to make a backup of the files on the webserver before you start over writing them.

#### **get**

This command allows you to transfer (get) files from the remote machine to your computer. If I needed to transfer the file 'about.html' to my computer (so i could edit it for example) then I would use this command:

```
get about.html
```

Again, if the file 'about.html' already existed on my local machine in the current working directory then it would be overwritten by the file I have just transferred from the remote computer so be careful.

There are some extra commands that might be useful....if you need to execute one of the commands **cd**, **ls**, **mkdir** on my computer from the terminal that is logged into the ftp session, then I put an exclamation mark (!) before the command like so:

```
!ls
```

The out put from this command will be a list of files from the working directory on my computer *not* the files from the remote machine.

Once you are happy with using ftp this way you may also wish to experiment with transferring multiple files at once, in which case you would use the **mput** and **mget** commands...one hint when doing this...actually lets make it two hints.

1. if you want to transfer an entire directories contents use the wildcard symbol '\*' ...this is commonly used in Linux to say that any combination of letters is acceptable for the filename. For example, if I have the following files on my local computer:

```
one.html
two.html
three.txt
```

and I type :

```
mput *.html
```

Then all files ending with \*.html will be transfered to the webserver, in this case that would be the files 'one.html' and 'two.html'. If I used the command :

```
mput *
```

then *all* files would be transfered. The wildcard can me used for all Linux commands that work with files and directories.

2. use the parameter 'i' if you are going to transfer lots of files, for example if I typed the command:

```
mput *
```

all files will be transfered but I will be asked by the terminal to verify that each file is to be transfered individually. However if I type the following:

```
mput -i *
```

Then I won't be prompted to type 'y' for each file to verify I want it to be transfered.

I suggest you experiment with using ftp from the command line a little bit to get familiar with it before you need to use it in a real world situation. Perhaps do this by creating some bogus files and transferring them between the webserver and your computer using directories that you use just for trying things out (so you don't run the risk of doing any damage to either file system). There are also more parameters to **ftp** that you may need to know depending on the configuration of the ftp server you are working with, so perhaps also consult the manual pages (**man ftp**) to learn more.

## INSTALLING MuSE

Now we get to the real thing - live streaming! So this assumes you want to set up a live stream using a live audio input from your soundcard...if you wish to stream from playlists please see the section later on in this manual dedicated to that issue.

The first thing you need to do is of course, to install **MuSE**.

As started earlier, you could use the package management system that comes with your particular flavour of Linux. **MuSE** is nicely supported for many varieties of Linux and there are packages online from the Debian and Gentoo systems. But we are going to do this the generic way...installing from source. So please visit <http://muse.dyne.org> and download the latest source distribution of **MuSE**, at the time of writing the latest version is 0.9, so I will download the **MuSE-0.9.tar.gz** file

Next up, once it has downloaded, use the **cd** command to change working directories so you are in the same directory as your downloaded source. Forgot where you put it? A common error - use **slocate** or **locate** to find the file, you may have to do a **updatedb** first. For example, I would type

```
slocate MuSE-0.9.tar.gz
```

and this would output to the terminal where my file is located and I would then use **cd** to get there.

Okedoke...now unpack the archive by typing **tar zxvf** followed by the name of the file you just downloaded, I would type the following:

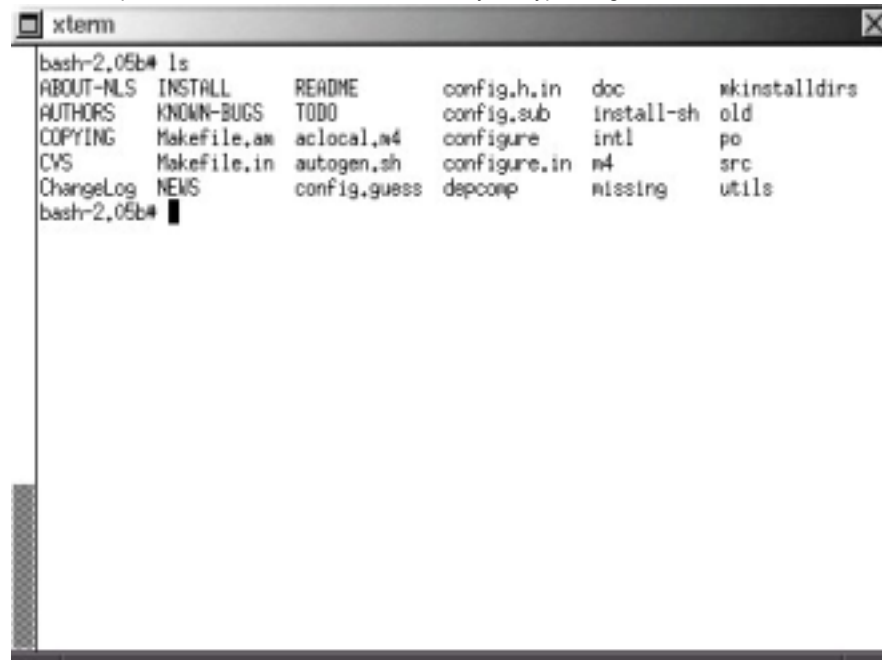
```
tar zxvf MuSE-0.9.tar.gz
```

And the file would unpack nicely to a new directory within the directory I am currently in. Typing a **ls** will show this. In my case there is a new directory called 'MuSE-0.9'. This directory could be named differently to what you have, depending on what version of the software you downloaded.

Now **cd** to the new directory. Again in my case I would type:

```
cd MuSE-0.9
```

This would place me in the **MuSE** source directory. If I type **ls** I get this:

A terminal window titled 'xterm' showing the output of the 'ls' command. The output lists various files and directories in a multi-column format. The files listed are: ABOUT-NLS, AUTHORS, COPYING, CVS, ChangeLog, NEWS, INSTALL, KNOWN-BUGS, Makefile.am, Makefile.in, README, TODO, aclocal.m4, autogen.sh, config.guess, config.h.in, config.sub, configure, configure.in, depcomp, doc, install-sh, intl, m4, missing, skinstalldirs, old, po, src, and utils. The prompt 'bash-2.05b#' is visible at the top and bottom of the terminal window.

```
bash-2.05b# ls
ABOUT-NLS  INSTALL      README      config.h.in  doc          skinstalldirs
AUTHORS     KNOWN-BUGS  TODO       config.sub   install-sh   old
COPYING    Makefile.am aclocal.m4  configure    intl         po
CVS        Makefile.in autogen.sh  configure.in  m4          src
ChangeLog  NEWS        config.guess  depcomp     missing     utils
bash-2.05b#
```

cool! these files are all the files you need (hopefully) to install **MuSE**

Now proceed by typing :

```
./configure
```

This should start the configure process. Now, having said you need all the files in this directory to install **MuSE**, this is indeed true but without some specific extra software **MuSE** is not much more than a simple mp3 player. So check the output of your configure script...it will look something like this:

```
config.status: creating src/resample/Makefile
config.status: creating intl/Makefile
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing default-1 commands
```

```
== MuSE build configuration =====
:: Building on i686-pc-linux-gnu (kernel Linux267)
:: Linking OggVorbis libraries for OGG encoding
:: Linking SndFile libraries for WAV/AIFF/VOC/etc encoding
:: Linking Lame libraries for MP3 encoding
:: Building the GTK-2 user interface
:: Building the NCURSES user interface
:: DEBUG informations OFF
:: PROFILING informations OFF (see man gprof)
:: CFLAGS = -pipe -D_REENTRANT -O3 -fomit-frame-pointer -march=i686 -ffast-
math
:: CXXFLAGS = -pipe -D_REENTRANT -O3 -fomit-frame-pointer -march=i686 -ffast-
mat
bash-2.05b#
:: LIBS = -lncurses
===== now type make, may the source be with you!
```

ok...the bit you need to look at is the end part which gives a nice summary of what libraries are installed that **MuSE** will work with. In the case of the output above the configure script has reported that I have all the possible libraries installed, ie I have **PggVorbis**, **SndFile**, **Lame**, **GTK-2**, and **NCURSES**

If I read the **README** file (use your favourite editor from the section above to open this file) in the **MuSE** directory, the file tells me this:

- LAME (optional)  
Lame can be installed but is no more needed, in particular libmp3lame and the header lame.h must be properly installed.
- OGG VORBIS (optional)  
You can compile and install libogg and libvorbis on your machine before compiling MuSE; the configure script will recognize them and include support for decoding and mixing of .ogg files.
- GTK+ (optional)  
if libgtk and all the related libraries are present, MuSE will compile the GTK+ graphic user interface for interactive use and additional fun.
- NCURSES (optional)  
if libncurses is present, MuSE will compile a text console interactive interface to be used into ASCII terminals.
- SNDFILE (optional)  
if libsndfile is present then you'll be able to play uncompressed sound files like wav, aiff, snd, voc, pvf, mat, au, sf etc.

So I can do everything possible with **MuSE** because I have all the libraries installed. However if the output from your configure script says something is missing then you need to consider whether you should install it or not. The basic minimum I would hope that you have installed is: **GTK**, **Lame**

If you have these two libraries installed you can at least have a graphical user interface on **MuSE** (provided by **GTK**) and you can stream MP3 (provided by the **Lame**) library. If you don't have these libraries get them from here:

#### **Lame**

<http://lame.sourceforge.net/>

#### **GTK**

<http://www.gtk.org/>

Note : I would be surprised if you didn't have **GTK** installed. Its possible, but unlikely as most distributions of Linux have this library as it contains many libraries needed by other softwares commonly found on Linux. If you don't have **GTK** it can be very tricky to install so I would recommend you get some help for this.

#### **Ogg**

If you also want to stream Ogg then you must first install libogg and libvorbis:

<http://www.xiph.org/ogg/vorbis/index.html>

If you need any of these libraries and the configure script says you don't have them installed download them first, run through the install process as outlined in the section above, and after each install run the **ldconfig** command to update the library database. Then once you have done this for each one, start again with the **./configure** process in this section.

Now, lets assume your configure script now gives you the output you want, indicating that all the required libraries are installed. Cool!

Now you just want to type the **make** command as so:

```
make
```

and finally type:

```
make install
```

If all has gone well you should just need to type :

```
muse
```

and the application will open as so:



## LIVE STREAMING WITH MuSE

Alright...now the real business...streaming. Now for this section I am going to make the assumption that you already have access to a streaming server and that someone has given you the details. Specifically you need to know:

1. The hostname (IP address or URL) of the streaming server
2. The login type of server being used (shoutcast/ Icecast / Icecast2 etc)
3. The password of the server
4. The mountpoint you should use (more on this later)
5. The port you will stream to (usually 8000)

Now...ask the person that administrate the streaming server for this information. If on the otherhand you intend to run your own server, as explained later in this manual, then you will get these details yourself.

So...onwards..type the following:

```
muse
```

and you should see something that looks like this:



Now at the top we have a toolbar:



Lets start from the left...



the first button is the button that opens the configuration panel. It is in this section that you enter the information such as that outlined above (hostname, port etc) as well as the format (bitrate, ogg/mp3 etc) of the stream. This is the part of the application that determines most whether you stream or not, so we will go over this in detail in a moment.





This button adds a new channel to **MuSE**, you won't need this until you are more experienced with **MuSE**.



This button means that we want to stream using the input from the sound card, you will want to turn this on, but we will get to that soon.



This button means you want to stream using playlists, this is covered in a later section and for live streaming using the sound card you will want to turn this button off.

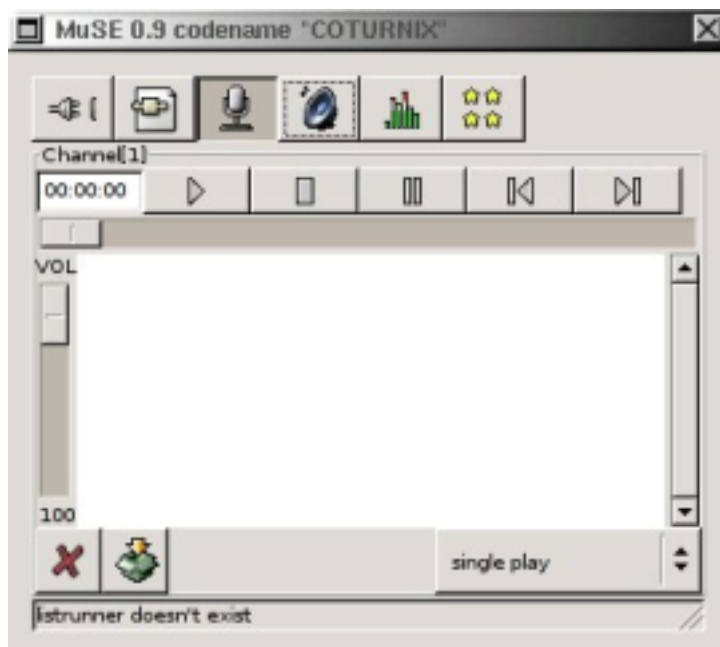


This button opens the volume meters for **MuSE** so you can monitor the volume of the outgoing stream.



This button opens the panel with the credits...check it out, its quite funny :)

Ok...to start, turn on the button with the microphone icon on it. When opened **MuSE** opens with the speaker button turned on by default and the microphone button turned off. To turn the microphone button on, first press the speaker button so it becomes 'unpressed' and then press the microphone button so it looks pressed, like so:



Next up....press the configuration button (first button) so you see this:

Right...this is important! We need to enter here all the information relevant to the stream we want to begin. Lets start with streaming MP3...so the panel that is to the front here is the panel for setting up a MP3 stream. if you want to do an Ogg stream then click on the panel 'Ogg/Vorbis Streaming' at the back, and jump forward in this section to where this is covered...for now we will just do MP3.

For now leave the top part of this form alone...the top sets the bitrate and quality of the outgoing stream, we will first just get the stream going then we will tweak this part a bit. ok...we have some information fields that need filling in, these all have to do with the server that you are streaming to. We will assume that you are using an **Icecast2** server. If you are using a different streaming server then everything will be the same except you will need to know what kind of **Login Type** the server is...I will let you know more about this when we come to fill in this field.

here is a field by field account:

#### Host

Enter here the URL or IP address of the server you will be streaming to. If you are running a server on your own machine (where you are also encoding from) then you need to enter 'localhost' in this field. Other wise you need to enter everything including the 'http://' section. For example, if my server was located at 'http://icecast.radioqualia.net' then I would enter exactky that into this field, ie:

```
http://icecast.radioqualia.net
```

The host information will be given to you by the person running the server.

#### Port

This is usually by default port 8000. The person running your server will give you this port number and you should enter it in this field, if they haven't given you a port number try 8000 first and if this doesn't work then call them and ask them for it.

#### Mount

A mount point is the unique point on the server that your stream will connect to. This means if you are running a streaming server and there are (for example) 20 people sending individual streams to the server then each can be identified by the mount point. It is also used in the playlist file you will create later so taht you can link to the stream from your webpage. It doesn't matter what the name of your mount point is, although sometimes the server administrator will give you a specific one to use. To make sure you

won't conflict with another stream don't use obvious mount point names like 'test' etc....also make sure your point point has a preceding '/' as indicated in the example screen shot above.

**Name**

The name that will appear in the media player of your listeners when they connect to your stream...using something more descriptive than just 'my stream' or some such.

**URL**

This is not the URL of the streaming server, its just a URL that sometimes also gets displayed in the clients player. Its for displaying the URL of the associated web page.

**Description**

Pretty much the same as the name field except you can fit in more information.

**Login Type**

This is the type of authentication your server uses, here is a table to help you choose the correct one:

SERVER TYPE	LOGIN TYPE
Darwin	icecast 1
Icecast 2	icecast 2
Icecast 1	icecast 1

**Pass**

Enter your password for the server here. This should be given to you by the system administrator for the streaming server.

Now start your stream by starting some audio from your mixing desk (or whatever you have plugged into your sound card), make sure your software mixer is turned up, then press 'connect' on this form and away you go!

## PUTTING IT ONLINE - CREATING PLAYLIST FILES

linking to live streams

MP3 and OGG streams are usually linked from a browser with an intermediary text file. These text files are often called 'playlists' and end with the suffix '.m3u' or '.pls'. The content of this file is simply a plain text line listing the location (URL) of the live stream.

If you don't use one of these files to link do this and you try and open a stream linked directly from a browser then the browser tries to download the file to your desktop. This is problematic as most browsers will just download the mp3 stream forever until the user terminates the download window of their browser. The user then has to open their mp3 player and relaunch the downloaded file to listen to it...not the easiest process, and certainly this doesn't give the feeling of a live stream experience!

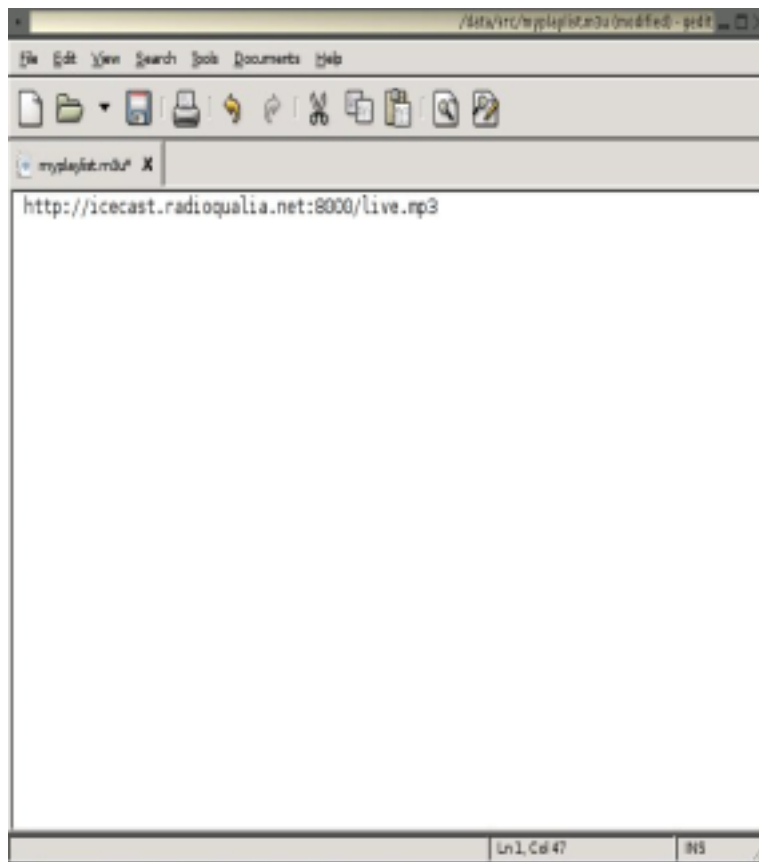
The playlist file is just a plain text file. You can include multiple links in the file separated by a line break. This has some advantage as the player that plays the live stream will first open the m3u file and then play the listed contents sequentially. This is very handy as sometimes there are server outages or net congestion that causes the player to terminate the connection to the streaming server without warning. In this case the player will read the next item in the playlist file and attempt to play that. If this is also a link to the same live stream then the player will re-connect and start playing the stream again (usually without the user even knowing something has gone wrong at all).

To create a playlist file simply open a text editor (see the section above) with the name of the playlist file following the command. For example if I want to create a playlist file called 'myplaylist.m3u' with the **Abiword** processor I simply type this in a terminal:

```
abiword myplaylist.m3u
```

The fact that the file 'myplaylist.m3u' didn't exist before simply means that **Abiword** (or any other text editor / word processor for that matter) will create the file when I have added some content (text) to the file and saved it.

So, once you have the file open in front of you with your favourite editor, you then complete one line with the full URL of your live stream. For example, if the location of my live stream was **http://icecast.radioqalia.net:8000/live.mp3** then I enter this URL directly into the text editor and save the file, like so (using **gedit**):



I could, by the way, use either suffix (**.pls** or **.m3u**) with the same format and it would make no difference.

Now, you need to transfer this playlist file to your webserver using an ftp application, and then link to it from your webpage (see next section).

its super hard to find info on playlist files like m3u...but heres a good overview:  
<http://gonze.com/playlists/playlist-format-survey.html>

also check out this definition of the m3u format:  
[http://www.schworak.com/programming/music/playlist\\_m3u.asp](http://www.schworak.com/programming/music/playlist_m3u.asp)

by the way...m3u is a non-proprietary format...on the other hand pls is a proprietary format (owned by winamp).....

## PUTTING IT ONLINE - LINKING PLAYLISTS FROM A WEBPAGE

Righto, never let it be said this manual didn't at least attempt to teach you all the basics associated with streaming! (I am assuming you have actually managed to turn the computer on and check the air in the tyres)

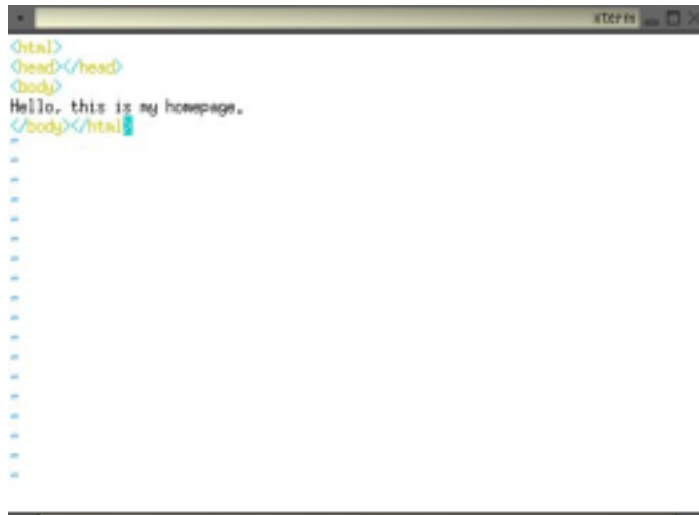
Linking in html is important to streaming otherwise your audience will be limited to those you have emailed the live URL to. Putting a link on a webpage connecting to your stream is important for allowing others that aren't on your mailing list, to get access to your live stream. To do this we link not directly to the live stream from the webpage for the reasons explained in the above section...instead we link to a intermediate file...the playlist file. Lets assume you have read the above section on creating a playlist file and you have made an example. I have done the same and my example file is called playlist.m3u and it contains one line (the link to my live stream):

```
http://icecast.radioqualia.net:8000/live.mp3
```

Now, lets assume I have a webpage on my webserver, and it is on this page that I need to include a link to the live stream. The first step I need to undertake is to download my file from the webserver using **ftp** (see the section above on ftp). If it is the front page of my website it will probably be called 'index.html'. Once I have downloaded this file I then open it up in my favourite text editor...for this example I will use **vim**, so the command I issue in the terminal is this:

```
vim index.html
```

and I would see my 'index.html' page opened, and all the HTML code will be displayed in my editor. For this example I will use a basic HTML page as my example file:

A screenshot of a terminal window showing the vim editor. The editor is displaying the contents of index.html. The code is as follows:

```
<html>
<head></head>
<body>
Hello, this is my homepage.
</body></html>
```

The cursor is positioned at the end of the last line. The terminal window title bar shows "vim".

Now I will make a link using basic HTML code to the playlist file. A link in HTML follows this syntax:

```
<a href='file'>open me!</a>
```

In the above line *file* should be replaced by the name of your playlist file. The 'open me!' part of the above line is the text that will be displayed in the browser that is associated with the link. So lets first add the link:



## THE POLITICS OF STREAMING : OGG vs MP3

Some Definitions

What really does 'open' mean in the context of codec's??

In this context an open codec means :

'Open' as in 'Source' : the source code is available for the compression algorithm to be accessed by developers

'Open' as in 'Patent Free' : development of the code must not be hindered by patent rights

'Open as in 'royalty free' : the codec must have the ability to be used either as an algorithm integrated into softwares (players, servers, encoders) as well the ability to be used as an encoding codec by content producers without an legal requirement that royalties should be paid now or at anytime in the future for using the codec.

What codecs are closed?

All codec's developed by RealNetworks and WindowsMedia and most codecs used by QuickTime.

What codec's are open?

Ogg Vorbis (audio codec - # )

VP3 (video codec - # )

What about MP3?

MP3 (short for MPEG - Layer 3 ) is a patented technology and has a schedule of royalty rates ( # ). Mp3licensing.com states that they will not charge "or private, non-commercial activities (e.g., home-entertainment, receiving broadcasts and creating a personal music library)" as long as the activities do not earn more than \$US 100,000 a year. However their patent protection policy has lead to various problems for free (as in GPL) MP3 softwares including Pluggger, CDEX, soloH, 8Hz, Blade, and Canna, furthermore these rules are company policy hence they can be changed at anytime.

And DivX?

DivX is a suite of codec's used most often for ripping DVD content. DivX is originally backward engineered from Microsoft technologies by a group called Project Mayo. However Project Mayo is now called DIVX and they are currently trying to patent the technology.

And, by the way, what is the Free Software, and what is Open Source??

The GPL (General Public Licence) is the Free Software Foundations licence for free software. This licence allows and protects certain freedoms, most fundamental of all is the ability to share code, distribute and change code (including code for codecs). The Open Source movement is a separate movement that believes that the term 'Free Software' doesn't appeal to business people and that 'Open Source' is a much more marketable term. The Free Software people generally don't like this.

What's the problem?

Problem 1 : Content Encryption

Most audio and video content on the web is encoded by proprietary codec's.

This means that almost all content encoded for artistic, cultural or independent media purposes is encrypted. Encrypted in the sense that the content has been converted to a closed file format which can only be 'decrypted' by media players that have the requisite licensed algorithms (codecs). Hence the owners of these algorithms (Thompson and Fraunhofer, Microsoft, Real Networks etc) own the key to the content. It is not a public key, its a closed proprietary key. You, the content producer, cannot unlock your encrypted file (content) unless you do so with the appropriate media player software. These player softwares are usually created by the software house that owns the codec or they are created by a third party who licence the key (codec) to unlock (decode) and play your file.

This may seem ok now, you probably haven't had any problems from this apart from the fact people may need to download a new player to experience your online streaming file. However is it enough to trust that all will always be ok. Consider, for example, if you have encoded (encrypted) some video content with a closed codec (lets take RealNetworks audio and video codec's for this example). For now you might have the key (algorithm) to 'unlock' the content and replay it in a player software (in this case the RealPlayer). However we can easily imagine a situation in 'X' years time where Real has crashed and burned and are no longer a technology provider. Where is your content now? Your content could well be in a encrypted file format with no licensed keys to open it. The codec may have gone down with the company and you may be left begging users to download the older players that you have found in some arcane archive somewhere on the net (ever tried looking for an old RealPlayer?), and who is to say the legal remains of (in this case) RealNetworks won't stop you from doing even that?. Distributing the software or its components (dlls / plugins etc) is illegal unless you have express permission from RealNetworks in this example and the right to stop you doing this may persist long after the companies death.

There is an interesting parallel to this with MAMEs (Multi-Arcade Machine Emulators). Arcade games from of all sorts, going right back to PONG, are available on the internet for download. However you require a software known as a MAME to interpret the file formats of these games, there are many of these emulators available but almost no more games (known as ROMS) can be retrieved easily from the net unless you know exactly where to go. This is different from a few years ago when you could easily get almost any ROM you wanted. However these sites have been systematically closed down by games companies protecting their interests. Many of the games companies that created the original ROMS have ceased to exist but the ROMS are now owned by other companies, and these new owners protect their interests by closing MAME ROM sites.

Could this happen with codec's?



It has already. Progressive Networks (formerly Real Networks) released an early video codec which they termed 'fractal video compression'. Many people used this codec as there were few choices available at the time for encoding video for streaming. However the codec is no longer supplied with the RealPlayer.

#### Problem 2 : Player Wars

Half the battle for free media on the internet is for the codec's, the other half is for the players. The most commonly distributed players – RealPlayer, Windows Media Player, and the QuickTime player support their own suite of codec's.

That means we have almost as many players as we have codec suites. How many people have RealPlayer, QuickTime Player, and WindowsMedia Player installed as well as perhaps XMMS and Winamp? Why can't you choose the player to play the content without the codec choosing it for you?

Unifying the codecs within a single interface wont happen because the marketing strategy of these companies relies on capturing an audience through their player interfaces. RealOne player, for example, tries to massage you into subscribing to their content channels - making you part of their 'revenue stream'.

Allowing new free software interfaces, with new interface and navigational paradigms, also won't happen (or is at least very difficult) because these companies control the licensing of the codecs to developers.

Additionally, until the open codec's are well supported in media players (download yours today!) then ironically media encoded in free codec's has a very limited audience.

Proprietary codec's make companies money. Hence they do not support open codec's because free codecs are competition.– the more open codec content there is, the less need you have for their players, the less money they make.

#### Problem 3 : Microsoft, Real, and Apple make money from your content.

WindowsMedia Player, QuickTime player, and the RealPlayer have advertising embedded in their interfaces. This advertising generates revenue for Microsoft et al. Because your content opens one of these players, more eyes get to see the embedded advertising which translates to \$\$\$ for these companies.

#### Problem 4 : What control do you have?

The question arises as to what control you have over the software you are using to stream. For example, the codecs determine the player, and the player is licensed to the end user. Do you know what the player licences permit and restrict? No?

Well here's an example from the Licence of the RealPlayer:

Paragraph 2(c) says :

"You may only use the Software for your private, non-commercial use. You may not use the Software in any way to provide, or as part of, any commercial service or application. Copies of content files, including, but not limited to songs and other audio recordings, which are downloaded or copied using the Software, and which are protected by the copyright laws or related laws of any jurisdiction, are for your own personal use only and may not be distributed to third parties or performed outside your normal circle of family and social acquaintances."

Any content displayed through the player is for "your own personal use only and may not be distributed to third parties or performed outside your normal circle of family and social acquaintances". So, for example, if you work at a Public Library and you want to make some public archives available for free then this is not permitted under the general licence of the RealPlayer.

What is the solution?

Open Source and Streaming

Its strange that this debate does not often enter the rosy world of the 'Open Source' idealists. Strangely Eric Raymond, president and co-founder of the Open Source Initiative, has a speech about The Cathedral and the Bazaar linked from his site in RealAudio format (#). This on the same site that he points out that "Unisys is shaking down websites that use GIFs for a \$5000 license fee" and links to the well known #

Why is this medium (online audio / video) not debated more often in the area of 'Open Source'?

It seems the potential consequences of closed and /or proprietary codec's have only dawned on a few, most importantly those at # where the development of the royalty free Ogg Vorbis audio codec takes place and # where the same people are trying to develop the VP3 open video codec.

So what can we do about this?

Well, if you are a programmer and have some time on your hands then you can contribute to the many projects aimed at countering the major technology providers in this field. One action could be to contribute to the sophisticated Icecast open streaming server project, or the various Xiph.org projects.

If you are a content producer you can take control of the interface making sure revenue streams are not built on the back of your content. If you stream content then consider linking to a player without embedded advertising or embed your stream in a webpage. This has the obvious secondary advantage of ensuring you control the aesthetic context that your work is displayed in, making sure your content is not surrounded and associated with ugly advertising.

However for most of us the biggest thing we can do is understand the issues, support those 'fighting the good fight' and prepare to convert our archives from proprietary codec's to 'fully open' (from the description of Ogg Vorbis (#) codec's. Its getting easier to do this, many players (not just those on Linux platforms but also popular players like Winamp) support Ogg Vorbis already and the list of supporting softwares for these free codec's is growing. And lastly, Get your free codec's now.

So, the question might become...which codec do I use?

First issues are:

1. mp3 is a patented technology
2. ogg is an open technology

The first question is a question about your position on 'open source' - does this matter to you on ideological grounds? If not then the answer still is 'which one do we use'...if you really want to make an ideological stand then the answer is simple - use ogg

ok...so, i dont know your answer to this so i will follow the argument through, ....if want to make a choice on ideological grounds then you need go no further! if you decide this isnt so much of an issue for you then it comes to a question of psuedo-legality

The question would be-'are you comfortable with the fact that legally you can be charged for using the mp3 format?'

this is not a question about 'open source' it is about royalties....thomson and fraunhofer own the patent to the mp3 format and they can charge you when you use mp3 as you have tacitly agreed to their licencing terms...see:

<http://mp3licensing.com/>

specifically check out:

<http://mp3licensing.com/royalty/index.html>

You wont pay of course if they ask...and the chances are they wont ask...infact they promise not to charge anyone earning under \$USD 100,000:

<http://mp3licensing.com/royalty/emd.html>

So chances are they wont ask you to pay...but also, they could change the licencing terms at will...so the question really is : "are you ok that you might one day end up illegally using the format even though you wont be caught?"

Lets say you dont care about the royalty issue...then the question still is "which codec will i use"....if you do care then problem solved! - use ogg

ok...again, if you have decided the potential to one day be illegally using a codec does bother you, then there is nothing further to discuss...however, if you dont care then the next question is: "do i care about audio quality?"

Well it doesnt matter which way you answer to this, i dont think it really effects the matter...there are arguments to and fro about which is a better codec but in general they are on a par...

So you still might be in the position of wondering which way to go...in which case the question is:

"does the fact that users may have to download a new codec before listening to your content bother you?"

If the answer to this is "no" then it doesnt matter which codec you use....if the answer is "yes" then mp3 has the edge as more players are capable of replaying mp3 than are capable of replaying ogg - although this advantage is rapidly diminishing as many popular players now include ogg decoders (eg. winamp)

....so i think thats the general question-tree i recommend....

the other option is of course - use both

## STREAMING FROM PLAYLISTS WITH MuSE

### step 1 Install Muse

MuSE runs under \*nix systems...please download the latest source code from <http://muse.dyne.org>

TO install, unpack the archive and install look at the section on installing software under Linux, but to refresh your memory:

```
tar zxvf muse.x.tar.gz
```

Then **cd** to the directory just created:

```
cd muse.x
```

then type the following:

```
./configure
```

```
make
```

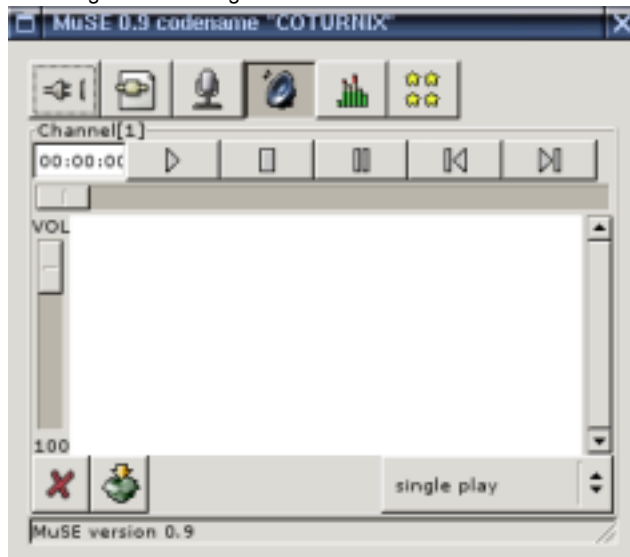
```
make install
```

### step 2 Start MuSE

type :

```
muse
```

You will get the following window:



ok!...so...if you see his all is well...if not then you probably need help ;-)

### step 3 Basic configuration

What you see before you is the mp3/ogg encoder 'MuSE' short for Multiple Streaming Engine

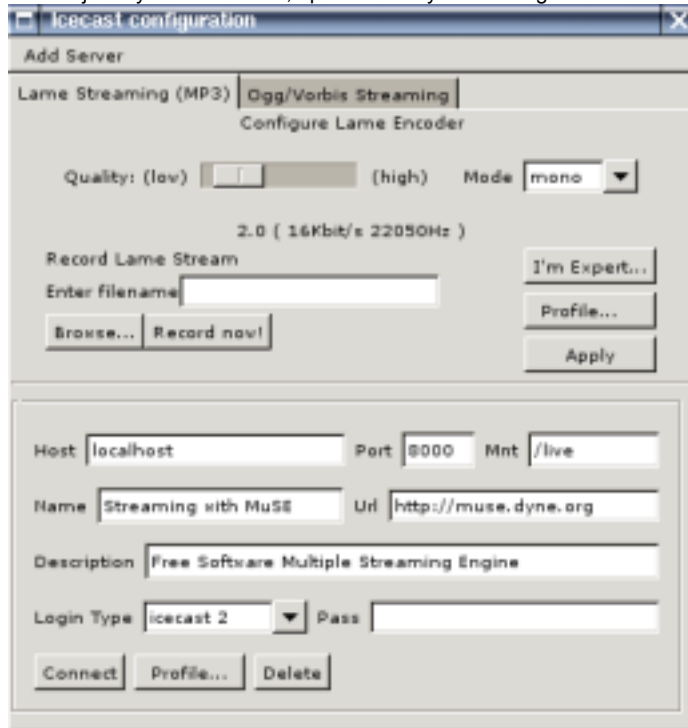
You really need to just get a bit familiar with the interface...its not too scary...you will see some top buttons:



these are, in order of appearance...

1. configuration panel button
2. extra channel button
3. line-in button
4. playlist button
5. VU meter button
6. credits button

for now just try the first button,...press it and you should get this:



#### step 4 Fields to fill

The basic idea is that you have got a streaming server somewhere that you can stream to...right? ;-)

The nice kind streaming guru that set this up for you has probably given you some information which would at least need to be:

1. address of the streaming server (host name)...this can be in the form of a 'url' or IP adress
2. port for the same server
3. password for the server

optionally they may have also stated a mountpoint

...well if you have this, then you have all you need to use MuSE to stream...it goes without saying you have an internet connection too...

you now need to obediently fill the correct fields with the correct information...starting with: HOST ...you can see this nice big field right? In the above picture the field has 'localhost' written in it...you don't want that, what you do want is to put the location of the streaming server in this field. If, for example, your streaming server was located at 'icecast.rq.net' then you would put exactly that in the field...do not put in a leading 'http://' or a trailing '/'...ok?

so then put the port number in the Port field, and the password in the Pass field...not too tricky so far....

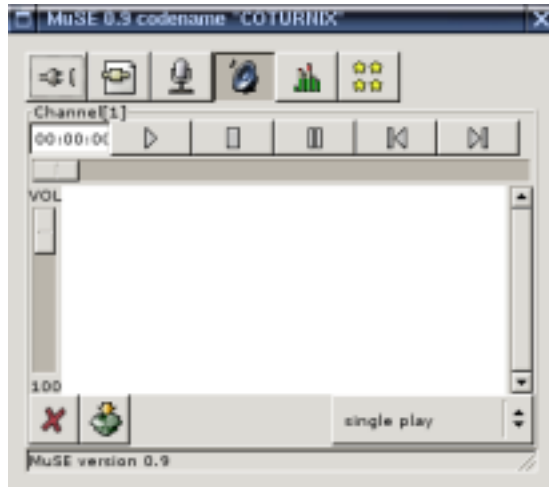
The other fields on this page are just because those kookie Italian coders like to put lots of pretty fields in there for looks ;-) (more on these later)

You might also want to be bothered with the Login Type drop-down box. You will need to choose the kind of server you are using here...I am just going to assume that you are using an Icecast2 streaming server so just choose that option and leave it.

So that's it, you are now a bona fide streaming master...but maybe before bragging about it too much, you should look to the following:

#### step 5 Starting the stream

So, let's look at the original window again:



what you want to decide now is if you want to stream a live sound or a playlist...in simple terms,...do you want to stream the sound that you input into your soundcard, say with a microphone, mixing desk etc...or do you want to play pre-recorded audio (mp3s etc) If you want to stream a live sound from the soundcard, then press the button with the big 'microphone' on it.... You may need first to 'de-press' the speaker button if its already looking like it has been pressed (as in the picture above).

If you want to play pre-recorded material, then you will need to create a playlist with MuSE, which is easy...the first step is to make sure the big 'speaker' button is pressed down.

In the later example (creating a playlist) once you have pressed the speaker button, then 'right-click' on the white colored window space and choose 'add file'...then browse to a mp3 file on your harddisk (or any format audio file) and add it to the playlist Do this as many times as you want....then when you are ready we can start streaming :)

step 6 Really starting the stream...

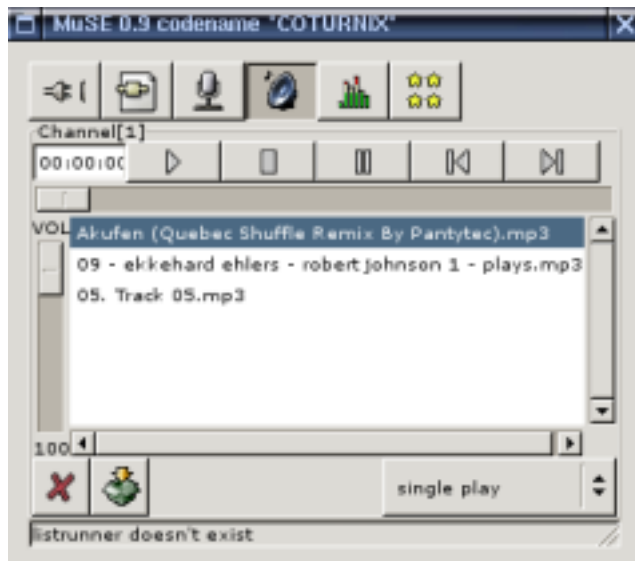
ok...if you have chosen the live sound option...then plug in whatever it is you want to use as a sound source into your sound card input and make some noise....

If you have chosen to play a playlist, then highlight one of the tracks in the MuSE window and press the 'play' button.

Then return to the 'configuration' window and press the big button marked Connect

thats it!!!!!!!

Some more pics with different options chosen:



MuSE with items loaded into the playlist.

## SETTING UP A STREAMING SERVER - ICECAST

As mentioned earlier, there is no magical 'server' that sits in a dark room which is specially created to send and receive streams. A server is just, at its most basic, a computer running a type of software called a 'daemon' or sometimes called a 'service'. For example, you turn your own computer into a 'web server' by installing and running a software like **Apache**. When running **Apache** will look after delivering all your web pages from your local machine.

Similarly you can run a streaming 'server' software like **Icecast** or **Darwin Streaming Server**, or **Litecast**, or **Windows Media Server**, or a **RealServer**. There are plenty of these softwares available and its just a matter of choosing the one to fit your needs, we will look at installing **Icecast2**.

I have chosen these **Icecast2** because it is easy to set up and use, its very stable, covered by the **GPL** licence and supports MP3 and Ogg.

The homepage for **Icecast** is at <http://www.icecast.org>

Right on the front page is a 'Download' link, click on this and you will be taken to the download section of their site (<http://www.icecast.org/download.php>).

At the time of writing the latest version is **2.0.2**, however its more than likely you will be working with a different version. No matter, unless you are reading this in 2040 the install process should pretty much be the same.

You might notice that there are different version of the software available, there are some versions available for Redhat for example, there is also a Windows version. The package we want is what is known as the **source package**. This is a single compressed downloadable file which contains all the installation files when you uncompress it. If you run a system with a package management system you could of course just type whatever package management command is appropriate for your system, for example:

Under Gentoo you would type in your terminal window:  
`emerge icecast`

Under Redhat you would type:  
`rpm -i icecast`

Under Debian you would type:  
`apt-get install icecast`

This is assuming you are online and everything on your system has been configured properly. The only issue doing it this way is that you then need to search for all the configuration files as the package management system will have put them somewhere on your system and not told you where. If you face this problem, then after installing **Icecast** with a package management system type in a terminal :

```
updatedb
```

This will update the locally stored database containing the locations of all the files on your system. This command might take a while to update your file database depending on the speed of your machine and the size of your file system. When the comamnd has finished updating the database type this in the terminal:

```
slocate icecast
```

Or sometimes, if this doesnt work try this command :

```
locate icecast
```

If **Icecast** is installed somewhere then you will get a scrolling output of all the files with the name 'icecast' in them, and a list of all directories with 'icecast' in their name...somewhere in there will be your **Icecast** configuration file. If the terminal window spits back a lot of text too fast then try either of these two commands:

```
slocate icecast | more
```

or

```
locate icecast | more
```

The '|' is known as a 'pipe' and it allows you to combine commands together. The above command line commands the search command **locate** or **slocate** with the command **more**. The **more** command lets you control output to the terminal so that you only get a page at a time and pressing the **spacebar** will display the next page. To find more out about **more** or **slocate** or **locate** type the following in your terminal window:

```
man more
```

or

```
man slocate
```

or

```
man locate
```

...now that's enough about package management hassles, one of the reasons I like using sources is that you can rely on knowing that the config files etc are going to be in one place if you need them. Also, it works across all types of Linux such as Redhat, Mandrake, Debian, Gentoo, Slackware etc etc. So download the source package, which is sometimes listed under 'all' in the 'Platform' sections of the Icecast site, but in anycase you will recognise it because source packages normally end in a **.tar.gz** suffix. In my example I will download the file **icecast-2.0.2.tar.gz**

Ok...so I hope you have downloaded this package to somewhere you can access...normally I download installation files to my home directory in a directory called **src**. You can put your files anywhere you like as long as you can access them, although be a bit careful as you don't want to clutter up the filesystem with installation files everywhere...I recommend putting it in your home directory. A home directory, by the way, is the directory with your username on it which is under the **/home** directory. To get to your home directory type the following in the terminal window:

```
cd /home/username
```

Where 'username' is your username, for example, my username is 'adam' so I would type:

```
cd /home/adam
```

The **cd** command means **C**hange **D**irectory and you use this command all the time to navigate around the file system. If you are in your home directory then look for a directory named 'src'. To do this type :

```
ls -al
```

This command **ls** means 'list', and typing this will show you all the files in your home directory. If you don't see a 'src' directory, then type the following:

```
mkdir src
```

The **mkdir** command means **M**ake **D**irectory, and typing this will create a directory called 'src', now you need to **cd** to that directory. This is where you should put the sources if you have nowhere better to put them. If you have not yet downloaded the sources then download them to this directory, if you have already downloaded the sources then you can move the sources to this directory using the **mv** command (assuming you know where the sources are)...

```
mv /root/icecast-2.0.2.tar.gz /home/username/src
```

In the above example I accidentally downloaded the sources to the /root directory and I am moving them to my home directory (of course, I would not use the above command but instead I would use my real username eg: 'mv /root/icecast-2.0.2.tar.gz /home/adam/src')

### **Decompressing the downloaded files**

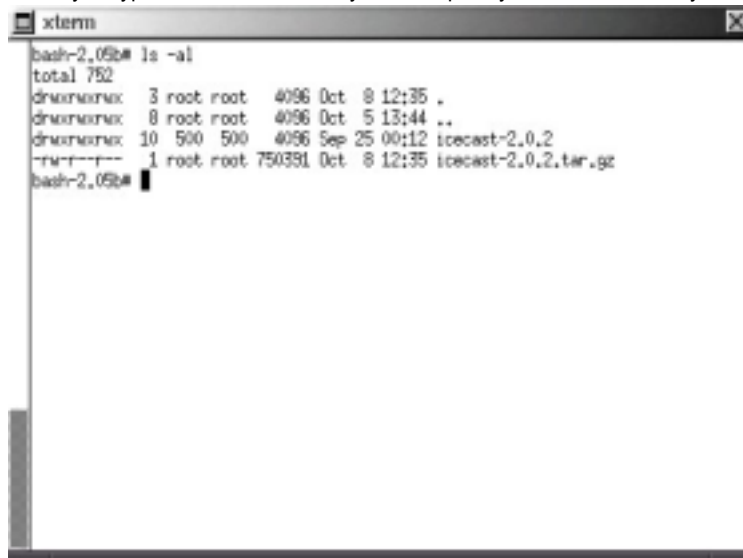
Now we will uncompress the file we have just downloaded...to do this type (assuming you have the 2.0.2 version, otherwise substitute the below file name for the one that you have):

```
tar -zxvf icecast-2.0.2.tar.gz
```



This will throw a whole lot of garbage into the terminal window, which is a list of all the files that have just been extracted from the archive. To learn more about the **tar** command use the **man** command.

Now if you type the **ls -al** command you will hopefully see a new directory:




```
xterm
bash-2.05b# ls -al
total 752
drwxrwxr-x 3 root root 4096 Oct  8 12:35 .
drwxrwxr-x 8 root root 4096 Oct  5 13:44 ..
drwxrwxr-x 10 500 500 4096 Sep 25 00:12 icecast-2.0.2
-rw-r--r-- 1 root root 750391 Oct  8 12:35 icecast-2.0.2.tar.gz
bash-2.05b#
```

so now you need to **cd** to the new directory. Then you need to type the following:

```
./configure
```

This is the standard configure command that you will pretty much use everytime you install something from source on Linux. If you are lucky you will see a whole lot of text scrolling through the terminal...this is the configure script checking that everything is ok on your computer before it allows you to install the software. If all goes well you should have a nice clean configure process with no errors and your terminal should end up looking something like this:



```
xterm
checking curl/curl.h usability... yes
checking curl/curl.h presence... yes
checking for curl/curl.h... yes
checking for libcurl... yes
checking whether CURLOPT_NOSIGNAL is declared... yes
configure: creating ./config.status
config.status: creating Makefile
config.status: creating conf/Makefile
config.status: creating debian/Makefile
config.status: creating src/Makefile
config.status: creating src/awl/Makefile
config.status: creating src/http/Makefile
config.status: creating src/thread/Makefile
config.status: creating src/log/Makefile
config.status: creating src/net/Makefile
config.status: creating src/timing/Makefile
config.status: creating doc/Makefile
config.status: creating web/Makefile
config.status: creating admin/Makefile
config.status: creating win32/Makefile
config.status: creating win32/res/Makefile
config.status: creating config.h
config.status: executing depfiles commands
bash-2.05b#
```

Now type the following:

```
make
```

The **make** command compiles a binary from the sources, basically what this means is that the configure script has worked out how to create the software for your system, then the **make** command uses this information to create the software. If the **make** process has gone well then you will end up with a nice clean terminal once again with no errors looking something like this:

```
xterm
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/adam/src/icecast-2.0.2/doc'
Making all in web
make[2]: Entering directory '/home/adam/src/icecast-2.0.2/web'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/adam/src/icecast-2.0.2/web'
Making all in admin
make[2]: Entering directory '/home/adam/src/icecast-2.0.2/admin'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/adam/src/icecast-2.0.2/admin'
Making all in win32
make[2]: Entering directory '/home/adam/src/icecast-2.0.2/win32'
Making all in res
make[3]: Entering directory '/home/adam/src/icecast-2.0.2/win32/res'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/adam/src/icecast-2.0.2/win32/res'
make[3]: Entering directory '/home/adam/src/icecast-2.0.2/win32'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/home/adam/src/icecast-2.0.2/win32'
make[2]: Leaving directory '/home/adam/src/icecast-2.0.2/win32'
make[2]: Entering directory '/home/adam/src/icecast-2.0.2'
make[2]: Leaving directory '/home/adam/src/icecast-2.0.2'
make[1]: Leaving directory '/home/adam/src/icecast-2.0.2'
bash-2.05b#
```

ok! so...if you have had problems and this is not what you see then you are going to need help...I'm sorry I can't say much here other than perhaps you need to go to your favourite search engine and start searching for answers. The best method I have found for doing this is to copy the exact error message you receive and type this into the search field of your favourite search engine and see what results you get. Chances are you will find someone else has had the same problem and has documented the solution or perhaps someone else has offered advice in a news group etc. In anycase, the problems that could occur are quite broad and its not possible to predict them all here.

If all has gone well then there is one more command to type:

```
make install
```

This command installs the compiled software in the appropriate place on your system. To check this has all gone well type the following:

```
icecast
```

This is the command that starts your newly installed **Icecast** server...if all is installed well you will see something in your terminal like this:

```
xterm
bash-2.05b# icecast
usage: icecast [-h -b -v] -c <file>
options:
  -c <file>    Specify configuration file
  -h           Display usage
  -v           Display version info
  -b           Run icecast in the background

bash-2.05b#
```

## USING ICECAST

Icecast is controlled completely from the command line. This is very handy if you need to run it on a remote server which is very often the case.

So to start Icecast the basic command is:

```
icecast -c icecast.xml
```

The trick is to find your 'icecast.xml' file. If you installed from source then you will find the file in the **Icecast** source directory within the **conf** directory. The file will be called 'icecast.xml.dist'. If its not there then run **updatedb** and then do a **locate** or **slocate** to try and find the file. So lets say I am in the **Icecast** source directory and the 'icecast.xml.dist' file is on the 'conf' directory, well maybe it would be easier if I moved it up to the directory I am working in and changed the name to 'icecast.xml'. I will use the **cp** command because then I leave a copy of the original just in case...so to do this I type:

```
cp conf/icecast.xml.dist icecast.xml
```

now I can run the command:

```
icecast -c icecast.xml
```

with luck it runs...the '-c' parameter here means 'config', so it is a way of telling the **icecast** command to use the config file 'icecast.xml'

actually sometimes I find that Icecast throws an error when it tries to create the log files, the error looks like this:

```
FATAL: could not open error logging
FATAL: could not open access logging
FATAL: Could not start logging
```

In this case I open my favourite text editor and start looking at the 'icecast.xml' file. I find these lines:

```
<logdir>/usr/local/var/log/icecast</logdir>
  <webroot>/usr/local/share/icecast/web</webroot>
  <adminroot>/usr/local/share/icecast/admin</adminroot>
```

So if I change these values to point to my home directory like so:

```
<logdir>/home/adam</logdir>
  <webroot>home/adam</webroot>
  <adminroot>/home/adam</adminroot>
```

You would of course use the name of your own directory. Now I find that **Icecast** works fine :)

Now you want to know the information like password etc so that you can plug them into **MUSE** and stream from your own encoder to your own server...well if you look at the 'icecast.xml' file again you will see these lines:

```
<authentication>
  <!-- Sources log in with username 'source' -->
  <source-password>hackme</source-password>
  <!-- Relays log in username 'relay' -->
  <relay-password>hackme</relay-password>

  <!-- Admin logs in with the username given below -->
  <admin-user>admin</admin-user>
  <admin-password>hackme</admin-password>
</authentication>
```

Don't worry about most of this, all you need to know is the default password is **hackme**. If you get a text editor and alter this password (the 'source-password') then you will need to stop **Icecast** (press **ctrl** and **c** simultaneously to do this) and then restart **Icecast** with the original command you use.

For the purposes of running **MuSE** with the **Icecast** server that you have just set up, use the following (all of which can be altered by editing the 'icecast.xml' file):

```
port : 8000  
password : hackme  
host : localhost
```

Then use any mount point you like, for example **/firststream** will work just fine.

If you do this then when you test the stream open **xmms** and enter into the 'open location' field (see section above) the following url:

```
http://localhost:8000/firststream
```

and you should hear whatever is coming through your soundcard.

## OTHER TOOLS

Well so far I have been quite exclusive when talking about streaming tools. I have discussed MuSE and Icecast only...bad me. There are a lot of tools available, but these two (Icecast and MuSE) are, in my opinion, some of the best for doing audio streaming under Linux. That's not to say there aren't some fantastic tools available, there are, but I recommend you get familiar with Linux and with MuSE and Icecast first before you start adventures in other softwares. But! For the time when you are ready, you might want to look at some of these goodies :) :

### PLAYERS

#### **zinf**

- many featured player but couldn't see the advantage over xmms (my preferred gui player) except that it has a built in streaming encoder/server.

<http://www.zinf.org/>

#### **soundplay**

- good if you have beOS (i thought it was for linux at first)...comes with built in streaming server....

<http://www.xs4all.nl/~marcone/soundplay.html>

#### **snackAmp**

- hugely featured gui player...almost too many features....has a built in webserver for remote control, and a built in streaming server....i would have used it for the project i was researching if it could work as a command line interface

<http://snackamp.sourceforge.net/>

#### **sonic-rainbow**

- looks nice and simple but i never tried it...plays video too

<http://sonic-rainbow.sourceforge.net/>

#### **impish**

- looked like a very sophisticated command line player but alas no crossfade and i couldn't get it to compile

[http://www.geocities.com/kman\\_can/](http://www.geocities.com/kman_can/)

#### **xmms**

- my favourite player if only because it does so much

<http://www.xmms.org>

#### **mplayer**

- more a video player but also support audio. you can run it with or without a gui

<http://www.mplayerhq.hu/>

### ENCODERS

#### **oggment**

- interesting tool which can encode a single audio source into a real stream and a ogg stream simultaneously....made by August Black (radio/software/media artist)...there are some other interesting tools on this site too

<http://oggment.sourceforge.net/ogg-real.php>

#### **oggcaster**

- plugin for xmms to stream ogg files to icecast2 ...looked interesting but i couldnt get it to compile...i think there was quite some activity on the development side, so it might be fixed soon...if doing encoding this way is your thing then dont forget the excellent xmms-oddcast plugin (for icecast2) and the liveice plugin (for icecast1 only) - both of which are available from the xmms

(<http://www.xmms.org>) website in the plugins directories

<http://xmms-oggcaster.sourceforge.net/>

#### **IceGenerator**

- streams ogg/mp3 without resampling...interesting idea but not quite what i needed so i ddnt try it...

[http://www.tortugalabs.it/modules.php?name=Downloads&d\\_op=viewdownload&cid=3](http://www.tortugalabs.it/modules.php?name=Downloads&d_op=viewdownload&cid=3)

#### **Soma**

a \_very\_ interesting set of tools...sound daemon, player, encoder and scheduler...ate a whole lot of my cpu and wasnt so nice with the crossfades, otherwise i might have used it...

<http://www.autistici.org/bakunin/soma/>

#### **darkice**

-a very light weight, easy to deploy command line live mp3 streamer

<http://darkice.sourceforge.net/>

**liveice**

- encoder for icecast1. Command line  
<http://star.arm.ac.uk/~spm/software/liveice.html>

**IceS**

- command line encoder for icecast1 and icecast2  
<http://svn.xiph.org/releases/ices/>

**SERVERS****gini**

- looks good...didn't try it  
<http://gini.sourceforge.net/>

**Oyez**

- python streaming server  
<http://ubertechnique.com/seth/oyez/doc/Oyez.html>

**oggserv**

- php based daemon  
<http://oggserv.sourceforge.net/>

**litestream**

- excellent streaming server with source clients and re-streamer...ultra stable  
<http://www.litestream.org>

**ample**

- small streaming server  
<http://ample.sf.net>

**edna**

- allows streaming of files stored on your 'server'  
<http://edna.sf.net>

**gnump3d**

- another streaming server, also supports ogg  
<http://www.gnump3d.org>

**Quicktime Darwin**

- Apples streaming server  
<http://developer.apple.com/darwin/projects/streaming/>