# Table of Contents

# Chapter 1: Introduction

## 1.1 Motivation and Objectives

Video LAN Client (VLC) is an open source application used to manipulate a variety of audio and video data. VLC is an incredibly versatile, highly portable multimedia player that supports a very large number of audio and video formats (detailed in Appendix D), as well as DVD and VCD playback, acquisition cards, and various streaming protocols. However, if simply being used to play audio and video locally, VLC has no real advantage over any other media player. VLC's real strength is that it can also be used as a server to stream in unicast and multicast on an IPv4 or IPv6 network everything that it is able to read, via UDP, RTP or HTTP. [3] This stream can also be transcoded on the fly, in real time, as it is being sent. Once the input stream has been received, it can be manipulated in various ways and output either to the screen, a file, or the network. [4]

Because of this versatility, VLC can be a valuable research tool for providing network performance measures. However, these performance measures are only relevant if the exact operation of the program is known and they can be taken in context. Unfortunately, reliable documentation describing the internal architecture of VLC is not readily available. It is difficult to best make use of the program's capabilities without detailed descriptions of its organization, implementation, and networking features.

The majority of documentation readily available does not address these concerns. Specific tutorials and user guides are available to answer common questions, almost always pertaining to the navigation of the user interface and the use of common functions. They rarely address the advanced functions of VLC and never describe its internal operations. The documentation that is available is written by the core VideoLAN team of approximately two dozen people, without any contribution from the many volunteer programmers who contribute significantly to the project. Because of this, the documentation is written once and unable to be maintained by the small administrative team, who are also responsible for overseeing all other aspects of the project. The documentation quickly becomes out of date and incomplete.

This thesis includes a detailed annotated bibliography listing all current available resources and describing their usefulness in order to provide a comprehensive reference for future investigation into VLC. From these sources, as much of the correct, relevant information as possible has been put together to form a cohesive report of the architecture of VLC. Emphasis has been placed on the modules and functions involved in encoding, decoding, and compressing data for transmission across a network and how specific features of VLC can best be used to take advantage of the way in which these tasks are performed. General advice on how to use the command line and properly form the syntax is also given, along with generic templates to aid the user in creating their own unique commands.

A significant amount of testing has been performed to demonstrate the results gathered in this report. As a demonstration of the problems with VLC's official documentation, the exact command line examples given are used as a starting point for testing the functions of VLC. These examples are then analyzed and developed into proper syntax. Test cases have been broken down into several categories based on the method of input and output being used. The results of each command are predicted and then tested. Test results are analyzed either simply by remarking whether or not the output reaches its destination or (in the case of transcoding) by gathering information about the output file (most importantly its container format, video codec, and audio codec) using the free, open source program MediaInfo.

The results of this thesis are useful not only to researchers interested in using VLC as a tool, but also to the open source community. However, this document is far from complete and there is always a need for more documentation, especially for a project that is constantly evolving.

## 1.2 Outline

The rest of this thesis is organized as follows:

- **Chapter 2** gives a brief summary of the problems that inherently arise in open source documentation and how these problems manifest in the focus project VLC. It then discusses the strengths and weaknesses of the specific documents available concerning VLC.
- **Chapter 3** is a description of the structure of VLC, specifically its modular design. These six main modules are key in understanding how to effectively use VLC, as the command line syntax directly reflects this modular layout.
- **Chapter 4** gives more in-depth answers to the basic questions about the use of VLC: What exactly are the default actions taken by VLC in the typical case when the user does not specify a course of action, and why would a user ever need or want to change these actions and customize them?
- **Chapter 5**, following from the previous chapters, gives the user the tools to customize VLC. These tools are presented in the form of general tips to remember when using VLC for any purpose and general syntax templates that can be used as a starting point to create custom commands.
- **Chapter 6** uses the templates given in Chapter 5 to give specific test cases of commands and the exact output generated by VLC. These examples are grouped based on input and output methods (file, device, or network). Incorrect examples are also given for demonstration purposes.
- **Chapter 7** evaluates the test cases presented in the previous chapter and summarizes the findings.
- **Chapter 8** presents conclusions and suggestions for future work.

# Chapter 2: A Look at Open Source Documentation

## 2.1 Introduction

In a project whose contributors number almost one hundred, span across nearly twenty countries, and consist of mostly volunteers who simply enjoy coding, it is incredibly challenging to create and maintain documentation that is correct, up-to-date, and consistent. The potential problems that can arise when trying to achieve this are endless.

In one scenario, the programmers also contribute documentation, at least for the modules or methods they are personally working on. Here, documents are all produced independently by different people, without any global decisions or agreements on terminology, style, and format. This leads to the misuse or inconsistent use of terminology, discrepancies between explanations and instructions across documents, and translation problems.

In the case of VLC, casual programmers (making up approximately 41% of the total attributed team) do not contribute to any of the documentation; it is all written by the core VideoLAN team of under two dozen people. While this allows for an increased consistency, it also means that documents are written once, become obsolete, and, due to time constraints, are never able to be rewritten. The "official" documentation (as available on the VideoLAN website) is thus terribly outdated, though the average user is unaware as it is not even indicated for what version a given document was written for, or if it was ever updated. Often documents are hastily thrown together, leaving a large percentage entirely blank, perhaps in the hopes that it would later be completed.

As with most open source projects, there is a greater need for volunteers capable of putting together clear documentation than there is for programmers. Very few people enjoy writing documentation in their spare time, but it is necessary for the success of a project.

Because of this, gathering the relevant information from VLC's sparse documentation is very difficult. This thesis elaborates on the useful information that was found and presents it in a clear, concise, manageable format. For anyone who wishes to consult VLC's available documentation, a complete survey of each resource's strengths and weaknesses is given below.

## 2.2 Survey of Literature

As a general overview of all available VLC documentation, it is essentially categorized How-To's. Arguably, having at least a few documents of this nature would be useful, if they were not all outdated and misleading. As a starting point to begin learning about

VLC, the command line examples as taken exactly from each of the official VLC documents that serve as user guides were input and run. Regrettably, the only reaction from VLC from any of these examples was to output a string of error messages (these are detailed in Chapter 5). This problem arises from misleading documentation. The command line interface is explicitly stated as being operating system independent, though this is not the case. A detailed explanation for these results is given in Chapter 7.

It is also very frustrating how much the different documents overlap. Of course, they were all written by the same small group of people, but the wording in two different documents is often so similar as to be almost exact. This caused quite a bit of confusion as it sometimes leads the reader to skip over a section, thinking that it has already been read, when there is really additional useful information. There are also cases of documents that appear to be entirely new because chapter titles were changed and rearranged, but it turns out to be the same as a previous document, or in another case, two documents merged together.

The majority of documentation is also woefully incomplete. It was very frustrating to open a document and find all the chapter headings listed and organized, but the body of the document was almost empty, consisting of entire chapters that contained only a "TO DO" reminder. Some documents also explicitly state that they are completely outdated.

Following are a few details about each specific document available regarding VLC. They have been ordered according to usefulness and their locations and reference information can be found in Appendix A.

### 2.2.1 VLC Streaming How-To

VLC Streaming How-To is easily the most useful VLC document. It is basically a streaming tutorial, but it contains explanations of basic concepts needed to understand streaming, step-by-step streaming examples using the GUI and the Wizard for each major operations system as well as advanced command line options (though this syntax for these examples does not work for every operating system if taken exactly as shown), and an example of each type of advanced streaming that VLC is capable of.

### 2.2.2 VideoLAN Streaming Features List

VLC's extensive features list is a necessity for using any of VLC's advanced features, especially if you plan on using the command line. The GUI will automatically direct you and only allow compatible options, but on the command line you will need this reference to know what features are supported in what operating system, what type of multiplexing is allowed when streaming to certain output methods, what audio and video codecs can be used in what container format, and what output formats can be used for the transcoding module.

### 2.2.3 VLC User guide

The VLC User Guide, despite its title, is actually the document that is least like a tutorial. It contains comprehensive lists of all modules available (video outputs, video filters, audio outputs, input modules, demultiplexers, interface, codec, OS support, and miscellaneous) and what options are enabled or disabled by default. The descriptions given for each of the modules could be better; most are just a single redundant sentence. This document also contains the best command line interface help section. The examples given are very basic, but the syntax given does work as is in most cases. It also gives the best explanation of how to actually use VLC's modules using the command line.

### 2.2.4 VLC Updates

This document gives details of the differences between each version of VLC in case certain features change or are not supported in certain versions. The changes are described relative to the previous version and are grouped into categories. It is unknown how complete this document really is as it was not used in this thesis.

### 2.2.5 VideoLAN Wiki pages

This site is an excellent idea in theory as it would be great to have all related documentation in one central place. However, in reality, it is just deceptive. The links listed cover all aspects of VLC; unfortunately almost all of them do not point to anything or point to blank documents. For example, *The Hacker's Guide to VLC* sounded incredibly useful, but so far only contains headers without any content. The documents that do exist are the official VideoLAN documentation copied to this new location. Currently, both copies do agree, though this could become a problem in the future. There are some useful documents here that do not exist anywhere else, including the *Installing VLC* guide.

### 2.2.6 VLC Knowledge Base (from the VideoLAN Wiki)

These pages provide general information about necessary concepts and terminology related to VLC; including codecs, file formats, protocols, hardware compatibility, servers, video output, and interfaces. It was useful for basic definitions, but as with all VideoLAN Wiki pages, there is a lot missing and left empty.

### 2.2.7 VLC Command Line Help (from the VideoLAN Wiki)

This document gives the full list of all possible command line arguments and brief descriptions of each. It is just the output of the complete command line help option, but it is easier to look at and browse than accessing the help from the command line.

### 2.2.8 VLC Play How-To

This document is a tutorial on the use of VLC. It contains a lot of information covered in other documents, specifically the VLC Streaming How-To and the VLC User Guide. It does give very in-depth instructions of how to use some of the more advanced features of VLC using the GUI, but that is basically its only contribution. Its Chapter 4 is taken directly from the VLC User Guide, but with some information missing.

### 2.2.9 VLC API Documentation

This document is the official API documentation for VLC. It is widely rumored to be incomplete, and is listed as incomplete on the VideoLAN site. The actual content is very insufficient and unprofessional. In essence, it is a How-To for codec and filter developers and only the information regarded as necessary for this purpose is given. It contains lists of certain methods but they are not explained well. A lot of potentially useful information is deemed unnecessary and intentionally left out.

### 2.2.10 VLC Frequently Asked Questions

Very few questions are actually addressed here and most have very simplified answers. The questions themselves seem very random, ranging from very general and vague (such as, VLC has strange behavior) to extremely advanced and specific (Where is VLC's config file?). The generic answer to almost every question about a problem with VLC is to reset preferences, delete the config file, and restart the program. A few legal concerns are addressed here that are helpful.

### 2.2.11 VideoLAN Play How-To/Advanced Use of VLC (from VideoLAN Wiki)

This document is an exact copy of Chapter 4: Advanced Use of VLC of the VLC Play How-To and is listed as being completely outdated.

### 2.2.12 VLC Doxygen Documentation

This is the documentation generated by Doxygen. It is nowhere near complete but may be somewhat useful to developers.

### 2.2.13 VLC Linux How-To Guide

All the chapters of this document are copied exactly from either the VLC User Guide or the VLC Streaming How-To. It does not contribute any extra useful information, except that it gives extra instructions on how to perform actions in VLS, though VLS is no longer being developed or supported as its functionality has all been added to VLC and improved. Chapters are arranged differently to make this document even more confusing.

This document is an excellent example of the problems with open source documentation. It copies an existing official document, puts it in a new, clumsy format, leaving out or adding small amounts of information in the process. The changes made are minor enough to make it incorrect, inconsistent and confusing. Even if the changes made are to correct the original document, these changes should be brought to the attention of the authors so the original document can be updated, or at the very least highlighted in the new document and explicitly explained (for example, so the user knows a change was made to accommodate a certain version). Posting a new, slightly different document in a different location is just overloading users with even more irrelevant information to sift through.

# Chapter 3: The Structure of VLC

In order to really understand how to use VLC, it is essential to be familiar with its design. VLC uses a modular system, which allows new functions and formats to be easily added. It also allows greater flexibility in development and integration, an important design feature for a program that is developed concurrently by so many different people. Also, the command line syntax and graphical user interface are modeled around the modular system. Options are accessed by the module they are contained in.

Certain modules are not enabled by default, and must first be enabled before they can be accessed. Depending on what release of VLC is being used, certain modules may not be included with the program at all. The official release available from the VideoLAN website contains all available modules, but due to copyright infringements, certain releases of VLC packaged with certain open source operating systems do not contain any modules that include proprietary formats. It is recommended to use the current stable release from the official VideoLAN website to be sure access to all modules is available.

The main categories of modules are video and audio output, video filter, input, codec, demultiplexer, and stream output. [3]

## 3.1 Output Modules

The video and audio output modules enable the system to display video to the screen and output audio. The most common types of output are enabled automatically for each operating system and can handle the majority of output systems. [3]

## 3.2 Video Filter Modules

The video filter modules are always enabled, allowing the user to perform modifications on the rendered image, such as transforming or rotating the video, inverting color, cutting the video into several split windows, adjusting contrast and brightness, cropping, cloning, or distorting the image. [3]

To use video filters when outputting to a screen, see the Video Filters section of Appendix D for the syntax and keywords to use.

## 3.3 Input Modules

Input modules allow VLC to read its input streams from different sources. DVD input, Video CD input, Audio CD input, and SLP input are enabled by default and HTTP, FTP, UDP, MMS and file input are always enabled. There are a few other less common input modules that are disabled by default, such as input from acquisition cards. [3]

Input modules are specified by the input stream. If the input stream is a file name, the file module will be used. This module is a bit different from the others in that VLC is not detecting the best option to use; there is only one possible correct module and it must be specified by the user so VLC knows where to access the input stream.

The proper syntax to access each type of input is described in Chapter 5, section 5.3.4.

## 3.4 Codec Modules

Codec modules add support for compression formats, so VLC is able to decompress streams in order to read them and compress streams into another format when transcoding. [3]

To force certain codecs to be used in priority when opening an input stream, see the Video and Audio Codec sections in Appendix D for the syntax and keywords to use.

## 3.5 Demultiplexer Modules

In a multimedia stream, the video and audio data streams are contained together in a container format. Demultiplexers separate the streams out of the container so they can be decoded and processed individually. All demultiplexer modules are always enabled. [3]

To force a certain demultiplexer to be used in priority when opening an input stream, see the Demultiplexer section in Appendix D for the syntax and keywords to use.

## 3.6 User Interface Modules

It is important to mention that various user interface modules exist that may be useful, notably the HTTP interface that allows VLC to be accessed and controlled from a web browser. However, they are not important to the focus of this thesis, sending and receiving data streams, and thus will not be covered in detail.

# 3.7 Stream Output Modules

The stream output modules allow any stream capable of being input and read by VLC to be output to a file or the network, as opposed to the traditional screen output. Different options and processing can be applied to the stream depending on the output method selected. Modules can also be chained and duplicated to combine features. The stream output modules currently available include *display*, *standard*, *transcode*, *rtp*, *es*, and *duplicate*. Each of these will be discussed in detail below.

## 3.7.1 Display Module

This module can be used to display the stream to the screen. This is most commonly used in conjunction with the *duplicate* module, so the stream can be viewed locally while being saved or streamed. [1]

## 3.7.2 Standard Module

This is the most versatile streaming module and will usually be used in every command chain. It is used to send the stream via any of the output modules or save the stream to a file. The three mandatory options for this module are the *access*, *mux*, and *dst* options. [1]

The *access* option sets the medium used to save or send the stream. These include saving the stream to a file, streaming to a UDP unicast or multicast address, streaming over HTTP (a secured SSL connection can be used if desired), streaming using the Microsoft MMS protocol, and streaming over RTP. However, to stream over RTP, it is a better idea to use the RTP module described later as it includes more options that are specific to RTP. [1]

The *mux* option sets the container format used for the resulting stream. Available options are the MPEG2/TS muxer, the MPEG2/PS muxer, the standard MPEG1 muxer, the ogg muxer, the Microsoft asf muxer (also has a special version for MMSH streaming), the Microsoft AVI muxer, and the multipart jpeg muxer. [1] Note that options selected must be compatible. Certain codecs are compatible with only certain container formats, and certain access methods require certain container formats. It is best to check with the VideoLAN Streaming Features [5] website as some formats have limited compatibilities. Available *mux* options and a complete description of each are listed by keyword in the Muxer section of Appendix D.

The *dst* option specifies the destination of the stream. This is entirely dependent on the *access* method selected. If saving to a file, the *dst* option would be the complete path of the file to save to. If streaming to the network, *dst* would be the unicast or multicast address and the port VLC should stream to. [1]

Other non-compulsory options are available to have VLC send Session Announcement Protocol (SAP) announcements.

### 3.7.3 Transcode Module

This module is used to re-encode the audio and video of a stream using different codecs and bit rates. Additional adjustments can also be made during transcoding, such as scaling, deinterlacing, cropping, or adjusting subtitles. Transcoding is always used in conjunction with another module in order to use the resulting output of the transcoded stream. Usually this is to save the transcoded stream to a file or stream it to the network. Depending on the input and output methods and the parameters set, transcoding can require a lot of CPU power. If the transcoding is being done locally (both input and output methods do not involve the network or a capture device), the transcoding can be done at the pace of the system. However, if the input or output methods do not allow pace control, transcoding is done dynamically in real time. [1]

Video options include specifying the compression format the video data of the input stream should be changed to, setting the bit rate of the transcoded video stream, setting the encoder to use to encode the video stream, setting the frame rate of the transcoded video, cropping the video, scaling the video, and deinterlacing interlaced video streams before encoding. [1]

Audio options include specifying the compression format the audio tracks of the input stream should be changed to, setting the bit rate of the transcoded audio stream, setting the encoder to use to encode the audio stream, setting the sample rate, setting the number of audio channels of the resulting audio stream, specifying the subtitle format, setting the number of threads to use to encode the streams, and syncing the audio to the video by dropping or duplicating video frames. [1]

Note that options selected must be compatible. Certain codecs are compatible with only certain container formats (which will need to be specified to send or save the transcoded stream), and certain access methods require certain container formats. It is best to check with the VideoLAN Streaming Features [5] website as some formats have limited compatibilities.

### 3.7.4 RTP Module

This module is used to stream over RTP. It also allows for RTSP support. Available options for this module are *dst*, the destination UDP address, *mux*, the container format to be used (note that only *ts* and *raw*, no encapsulation, are possible values for RTP streams), *ttl*, the TTL (Time to Live) of the sent UDP packets, *sdp*, how the stream's SDP (Session Description Protocol) file should be made available, and *port*, the UDP port used to send the first elementary stream. Separate *port* options are also available for

audio and video individually. There are also additional options for giving additional information about the stream, such as a name and description. [1]

### 3.7.5 Elementary Stream Module

The elementary stream module allows VLC to process the audio and video streams separately. This module has two separate *access*, *mux*, and *dst* options, one for video and one for audio. As with the standard module, the *access* options set the medium used to save or send the stream, the *mux* options select the container format for the streams, and the *dst* options specify the path or address of the destination based on the *access* methods. [1]

### 3.7.6 Duplicate Module

The duplicate module can be used to duplicate the stream and process it through several different chains. The only option this module takes is *dst*, but multiple *dst* options must be specified for the stream to actually be duplicated. Any of the stream output modules described earlier can be used as parameters of this option. A non-compulsory *select* option can also be set to duplicate only certain elementary streams. [1]

# Chapter 4: Customization

## 4.1 Default Settings and Actions of VLC

VLC is designed to detect the best choice of action for any file. It selects the best output module (this will depend on your operating system), the appropriate demultiplexer and decoder, and takes advantage of whatever hardware it can. A file will always be streamed and opened as is (in the current container and compression format) if it is not being transcoded.

The command line interface and graphical user interface are both available by default. The command line is the most powerful and flexible interface, allowing the user the greatest control over the program, and the graphical user interface is the most user-friendly, giving the user access to a Wizard and automating input, output, transcoding, and filters.

Most of VLC's options and modules are enabled by default and directly available. Those that are enabled by default are that way for a reason; they exist to take advantage of a certain resource if possible, and do not cause any harm if not. Therefore, there is no reason to ever disable these options, though it is always possible to.

For example, all of the *CPU* command line options, listed in Appendix E, are enabled by default to take advantage of any special hardware features that might be available. If these features are not supported, they are simply not used. Also, all demultiplexer modules are enabled by default to allow VLC to read as many container formats as possible. There would be no reason to disable any of these modules, as if a certain format is not required, it does no harm to still have access to it.

## 4.2 Why customize VLC?

The question that naturally follows from the previous section is, if VLC can detect the best output module, the appropriate demultiplexer and decoder, and takes advantage of whatever hardware it can, why would the user want to tamper with any of its options? To put it simply, VLC is not perfect. While it always makes an attempt to detect the best choices for a file it is opening, sometimes there are problems and these settings cannot be detected automatically. Also, there are advanced features of VLC that require certain options to be selected, such as for streaming and transcoding a file. [3]

In essence, the user does not really ever choose a demultiplexer or a decoder. However, VLC will allow you to set options to force certain ones to be used in priority. Therefore, VLC will attempt to use those demultiplexers/codecs first, and then check the rest for compatibility.

Choosing a multiplexer and encoder is necessary when you are transcoding (changing the encapsulation and compression) a file because you want to change the file's format. You can also choose these options during streaming if you want to transcode the file as it is being sent. Transcoding a file while streaming requires a bit more planning; the transformation must be one that can be done efficiently by your CPU because it is transcoding in real time as it sends the video. If possible, it is always better to transcode to a file first and then stream the transcoded file. This removes the time constraints and will increase quality and performance, especially if your CPU is slow.

Sometimes it is necessary to transcode a file a certain way because it is a more common format (can be opened by a wider variety of media players). VLC is often used to transcode less compatible file types (ones that are played by very few players) into a more common format. Sometimes this will even require that a codec module be enabled if the format is especially uncommon (it is supported by VLC but is disabled by default). This is especially important if the file is intended to be streamed, so the clients do not require a specific media player to receive the stream. Transcoding a file to a format that has a smaller bit rate may also be important if planning to stream a file, as it will be faster to send and improve performance.

In certain cases, it may be necessary to customize the video output module (using one of the output types that is usually disabled) if you have an obscure video card or output system. This applies equally to audio output.

To receive input from any type of acquisition card, the appropriate video input module must be enabled, as all of this type of input module is disabled by default.

Another common option is to select a different interface. VLC provides eight different user interfaces, each with its own strengths.

In the next section, I will describe how to access these options effectively using a standard, generic template that will easily lend itself to customization so any desired effect can be achieved.

# Chapter 5: General Syntax Guide

## 5.1 Introduction

The goal of this chapter is to present various guidelines and general examples to give the user a template to work from to create custom commands. Specific examples and their outcome will be examined in Chapter 6.

For the duration of this document, examples and instructions will be provided using only VLC's command line interface. This is for three main reasons: First, the command line instructions are supposedly common to all operating systems (this will be examined in detail later), unlike the graphical user interface (GUI) which is different for each platform. Second, the GUI is fairly intuitive, so an adept user (with the proper understanding of VLC's modules) should be able to easily navigate it. Third, the GUI is more likely to experience significant changes between versions, while the command line interface remains more consistent (changes are primarily due to the addition of features, not changing the syntax of current features). Finally, there are certain features that are only accessible from the command line.

Before restricting discussion entirely to the command line interface, there are a few things that should be noted about the GUI. VLC provides an excellent Streaming/Transcoding Wizard which is very helpful. It is well laid out and only allows the user to select compatible options. It is highly recommended, even for tasks that require a lot of customization. Available options are grouped by modules (as they are in Chapter 3) to make them easier to find and understand.

The following section gives general advice for using VLC's command line interface.

## 5.2 General Advice

There are a few important things to make note of when using VLC and the command line interface.

### 5.2.1 Troubleshooting

As an initial troubleshooting step whenever VLC is exhibiting strange behaviour, especially all of the sudden, the best thing to do is reset VLC's preferences (it has not yet been discovered if this is possible using the command line and may have to be done using the GUI), delete the configuration file, and restart the program [4].

It is always beneficial to add the following option to increase the verbosity of VLC's messages:

```
-vvv
```

It does not affect any other options and is useful for getting feedback about why a command does not work.

### 5.2.2 Operating System-Specific Advice

While command line syntax is consistent across the different operating systems, Windows users will have to use the following syntax:

```
--option-name="value"
```

instead of:

```
--option-name=value
```

which is allowed by the other operating systems. [2] This is to avoid problems with certain characters that Windows recognizes as special characters. Placing the value inside quotation marks forces it to be taken as the value.

Users with a UNIX based operating system will have to enclose their stream output chains in single quotes for the command to be recognized.

Certain commands that deal with modifying VLC's modules under UNIX based operating systems require root access.

### 5.2.3 Video Output

Video filters refer to any postprocessing done to the video immediately before it is displayed. Therefore, video filters only apply to the on screen display and thus cannot be streamed. They only format how your video is displayed once VLC receives it [2]. In order for filters to take effect on the client machine, they would have to be set on the client machine.

The user may wish to consider using the GUI to apply video filters, as it is very straightforward and is a hassle to do on the command line. By opening the *Settings* menu and selecting *Extended GUI*, all available video filters are conveniently laid out on the main interface so the direct consequences of adding and adjusting filters can be seen immediately.

To play the playlist items continuously, add the

```
--loop
```

option.

# 5.3 Syntax

These examples follow directly from the modules and options described in Chapter 3.

### 5.3.1 Notation Conventions

For the remainder of this thesis, regular expression notation will be used to describe generic commands.

Open and close square brackets ( “[]” ) are used to enclose portions of a command that are optional.

The Kleene star (“*”) is used to denote portions of a command that may appear any number of times, or not at all.

### 5.3.2 All Purpose Command Template

This is the general syntax template that should be used for any command [1]:

```
vlc input_stream --sout=#module{option=parameter[{parameter-
optionlist}][,option2=parameter]*}[:module{option1=parameter,opt
ion2=parameter}]*
```

This template will be easier to understand VLC’s modules and the options associated with them have been described.

A condensed form of this syntax can also be used, though it is not as flexible and cannot be passed as many options [1]. *Access*, *mux*, and *dst* are the options are described in Chapter 3, section 3.7.2.

```
vlc input_stream --sout access/mux:dst
```

An alternative syntax template is also given, but it should never be used. It does not work as expected and does not have an equivalent directive to duplicate output, as is easily implemented using the syntax above and the duplicate directive. This alternative syntax is shown below:

```
vlc input_stream --sout-module1-option1=parameter [--sout-
module1-option2=parameter]* [--sout-module2-option1=parameter --
sout-module2-option2=parameter]*
```

### 5.3.3 Global Versus Item Specific Options

Options can either be set as global (set for the duration of the program and apply to all streams) or item-specific (apply only to the stream directly before it and override any previous global settings).

Global options are set with the following syntax:

```
--option
```

Item specific options are slightly different:

```
:option
```

Global options can be specified in any order (as they will be applied to all streams anyway) but item-specific options must be placed immediately after the applicable stream. [2]

Examples of using item specific and global options:

```
vlc input_stream1 :option1 input_stream2
```

This command will apply *option1* only to input_stream1 and input_stream2 will remain default. [2]

```
vlc --option1 input_stream1 input_stream2
```

Alternatively, this command will apply *option1* to both input streams. [2]

### 5.3.4 Input Stream

Multiple input streams can be specified on the command line. They will be enqueued in the playlist in the order they were listed. An input stream is specified using the following syntax.

| | |
|---|---|
| File | [file://]file<br>file is the complete path of the file to open |
| HTTP stream | http://ip:port/file |
| FTP stream | ftp://ip:port/file |
| MMS stream | mms://ip:port/file |
| Screen capture | screen:// |
| DVD | [dvd://][devicepath]<br>devicepath is the DVD drive letter, followed by a colon |
| Simple DVD (no menus) | [dvdsimple://][devicepath]<br>devicepath is the DVD drive letter, followed by a colon |

| | |
|---|---|
| VCD | [vcd://][devicepath]<br>devicepath is the CD drive letter, followed by a colon |
| Audio CD | [cdda://][devicepath]<br>devicepath is the CD drive letter, followed by a colon |
| UDP stream [4] | udp:[[<source address>]@[<bind address>][:<bind port>]] |

### 5.3.5 Module Selection

For the additional modules, VLC will usually choose the most appropriate option available. To force a specific choice, use the following syntax.

```
--intf module
```

This option selects the interface module that will be launched as the main interface.

```
--extraintf module
```

This option selects select extra interface module that will be launched in addition to the main interface. This is useful for accessing VLC's control interfaces.

```
--aout module
```

This option selects the audio output module.

```
--vout module
--V module
```

Both of these options (alternative syntax) select the video output module. [2]

In the following chapter, specific command line input examples for various combinations of input and output types will be given to demonstrate the correctness of the syntax given in this chapter. They will also serve as more concrete examples to give a better understanding of using VLC's command line interface.

# Chapter 6: Implementation

The exact commands used to test and demonstrate VLC's modules will now be detailed, along with the precise specifications required to reproduce all of these test cases.

## 6.1 Environment Specifications

The following details the specifications of the systems used as client and server, and their connection.

**Client Computer**

| | |
|---|---|
| Operations System: | Windows XP Professional SP3 |
| Processor: | AMD Athlon Processor, 1.00 GHz |
| Memory: | 1 GB of RAM |
| VLC Release: | 0.8.6a |

**Server Computer**

| | |
|---|---|
| Operating System: | Windows XP Professional SP2 |
| Processor: | AMD Athlon 64 Processor, 1.81 GHz |
| Memory: | 512 MB of RAM |
| VLC Release: | 0.8.6a |

**Network Connection**

A local area network connected through a 100 megabit full duplex unmonitored switch

To test the output of transcoding operations, the open source program MediaInfo (v0.7.4.5) was used. It displays container and codec information for each audio and video stream, along with specifics of each audio and video stream and is available here: http://mediainfo.sourceforge.net/en

The specifications of the input files (as provided by MediaInfo) are available in Appendix C.

## 6.2 Test Cases

The test cases have been grouped by input and output methods. The three standard input and output methods are file, device, and network. The test cases cover all combinations of these three methods that VLC supports.

A brief description of the intended purpose of each command is given, then the exact command is given as it was input, and finally a description of the results, including a general description of the output (only applicable if the output device is the screen), any error messages produced by VLC, specifications of the output as determined by

MediaInfo (only applicable if the output method is file), and an overall decision of whether or not is was successful.

Though it is not shown for each command, the option

```
-vvv
```

is added to the beginning of each command that was input to increase the verbosity of VLC's messages.

### 6.2.1 File to Screen

General syntax form:

```
vlc my_file
```

VLC recognizes the file type automatically and opens the file. If it does not recognize your file type, you can tell VLC what codec to use in priority with the

```
--codec codec_module
```

option. For example, to play an avi file, use:

```
vlc --codec avi C:\VLCTest\Sliders4.avi
```

However, even if the codec you specify is wrong, VLC will check that codec first and then find the proper one, so it can still open the file. Therefore, the syntax:

```
vlc --codec ffmpeg C:\VLCTest\Sliders4.avi
```

will still open the file and play it correctly.

### 6.2.2 Device (DVD) to Screen

General syntax:

```
vlc dvd://[device]
vlc dvdsimple://[device]
```

If your DVD drive uses the default path ( /dev/dvd on GNU/Linux operating system or D: on Windows operating system) it is not necessary to specify it.

To open a DVD with menus:

```
vlc dvd://
```

```
vlc dvd://D:
```

Both of these commands open the DVD to the main menu.

To open a DVD without menus (begin playing the DVD immediately):

```
vlc dvdsimple://

vlc dvdsimple://D:
```

Both of these commands begin playing the content of the DVD automatically without recognizing any menus.

### 6.2.3 File to File (transcoding)

General syntax:

```
vlc input_stream --
sout=#transcode{vcodec=<string>,vb=<string>,acodec=<string>,ab=<s
tring>}:duplicate{dst=std{access=file,mux=<string>,dst=<string>}}

vlc input_stream --
sout=#transcode{vcodec=<string>,vb=<string>,acodec=<string>,ab=<s
tring>,access=<string>,dst=<string>}
```

There are additional options available for this module, but these are the most commonly used.

This is the exact syntax given in VLC's documentation to transcode a file to an example format [1]:

```
vlc input_stream --sout
'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}'
```

This was testing using the DVD drive as the input stream as follows:

```
vlc dvdsimple://D: --sout
'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}'
```

The layout of this syntax is incorrect (the single quotes are unnecessary, an equals sign is missing between *sout* and *#transcode*, and no destination is specified) and the following error message was produced:

```
stream_out_standard error: no access _and_ no muxer (fatal error)
main error: stream chain failed for
`std{mux="",access="",dst="'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}'"}'
main error: cannot start stream output instance, aborting
```

In the following properly formatted example, input is taken from the DVD drive and output to a file. A random file extension ".xyz" is used to test whether the file extension

needs to be in agreement with the container format indicated by the multiplexer (the *mux* option), as specified in VLC's official documentation. [4]

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=mp4v,vb=1024,acodec=mpga,ab=192}:duplicate
{dst=std{access=file,mux=ts,dst="C:\KATETEST.xyz"}}
```

The output was saved to the proper file. The file extension is not required to match the container format as VLC was still able to play this file properly as is. This is the output of MediaInfo for the newly transcoded file. All specifications are in agreement with the given transcoded file, except for a slight adjustment in the video bit rate. The desired bit rate was 1024 but is reported as being 934.

```
General
Complete name            : C:\VLCTest\KATETEST.xyz
Format                   : MPEG-4 Transport
Format/Family            : MPEG-4
File size                : 11.3 MiB
PlayTime                 : 1mn 22s
Bit rate                 : 1148 Kbps

Video
Codec                    : MPEG-4 Video
PlayTime                 : 1mn 22s
Bit rate                 : 934 Kbps
Width                    : 720 pixels
Height                   : 480 pixels
Aspect ratio             : 4/3
Standard                 : NTSC
Resolution               : 8 bits
Interlacement            : Progressive

Audio #0
Codec                    : MPEG-1 Audio layer 2
PlayTime                 : 1mn 22s
Bit rate                 : 192 Kbps
Bit rate mode            : CBR
Channel(s)               : 2 channels
Sampling rate            : 48 KHz
Resolution               : 16 bits
Video0 delay             : -33ms
```

After executing the exact same statement, but changing the video bit rate to 800 (*vb=800*), this is the output of MediaInfo for the newly transcoded file. Again, all specifications are in agreement with the given transcoded file, except for a slight adjustment in the video bit rate. The desired video bit rate was 800 but is reported as being 742. This suggests that there are certain bit rates that are standard, and VLC adjusts the input bit rate to correspond to the closest standard bit rate.

```
General #0
Complete name            : C:\VLCTest\KATETEST.xyz
Format                   : MPEG-4 Transport
Format/Family            : MPEG-4
```

```
      File size           : 9.36 MiB
      PlayTime            : 1mn 22s
      Bit rate            : 952 Kbps

Video #0
      Codec               : MPEG-4 Video
      PlayTime            : 1mn 22s
      Bit rate            : 742 Kbps
      Width               : 720 pixels
      Height              : 480 pixels
      Aspect ratio        : 4/3
      Standard            : NTSC
      Resolution          : 8 bits
      Interlacement       : Progressive

Audio #0
      Codec               : MPEG-1 Audio layer 2
      PlayTime            : 1mn 22s
      Bit rate            : 192 Kbps
      Bit rate mode       : CBR
      Channel(s)          : 2 channels
      Sampling rate       : 48 KHz
      Resolution          : 16 bits
      Video0 delay        : -33ms
```

The following transcoding example is using a video codec that is incompatible with the container format, as shown on the VideoLAN Streaming Features website [5]. This file should not transcode properly, or at least the video should not transcode properly. The following command was input:

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=h264,vb=512,acodec=mp3,ab=250}:duplicate{d
st=std{access=file,mux=ogg,dst="C:\VLCTest\KATETESTogg.xyz"}}
```

The file was saved to the proper location and did play in VLC. The correct audio was output but no video was displayed. This error message was output during transcoding:

main error: cannot add this stream
stream_out_transcode error: cannot add this stream

This is the output of MediaInfo for the newly transcoded file. Note that the audio and general information are all as specified for the transcoded file:

```
General #0
      Complete name       : C:\VLCTest\KATETESTogg.xyz
      Format              : Ogg
      File size           : 2.58 MiB
      PlayTime            : 1mn 24s
      Bit rate            : 256 Kbps

Audio #0
      Codec               : MPEG-1/2 L3
      Codec/Family        : MPEG-1
      Codec/Info          : MPEG-1 or 2 layer 3
      Bit rate            : 256 Kbps
```

```
      Channel(s)           : 2 channels
      Sampling rate        : 48 KHz
```

The following transcoding example uses compatible codecs and container format and should transcode without and errors. The following command was input:

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=theo,vb=512,acodec=mp3,ab=192}:duplicate{d
st=std{access=file,mux=ogg,dst="C:\VLCTest\KATETESTogg2.xyz"}}
```

The file was saved to the proper location, though transcoding was significantly slower than in previous tests. The newly transcoded file plays properly in VLC (both video and audio). This is the output of MediaInfo for the newly transcoded file and corresponds correctly to the specifications given for the transcoded file:

```
General #0
      Complete name        : C:\VLCTest\KATETESTogg2.xyz
      Format               : Ogg
      File size            : 6.61 MiB

Video #0
      Codec                : Theora
      Width                : 720 pixels
      Height               : 480 pixels
      Aspect ratio         : 4/3
      Frame rate           : 29.970 fps
      Standard             : NTSC

Audio #0
      Codec                : MPEG-1/2 L3
      Codec/Family         : MPEG-1
      Codec/Info           : MPEG-1 or 2 layer 3
      Bit rate             : 192 Kbps
      Channel(s)           : 2 channels
      Sampling rate        : 48 KHz
```

The following three commands were input separately in an attempt to test the alternative syntax method described in VLC's official documentation [1]. All three commands are meant to transcode the DVD input stream and save it to a file. Each command has slight variations in an attempt to tweak it to receive the proper results.

```
vlc dvdsimple://D: --sout-transcode-vcodec=mp4v --sout-
transcode-vb=1024 --sout-transcode-acodec=mpga --sout-transcode-
ab=192 --sout-standard-access=file --sout-standard-mux=ts --
sout-standard-dst="C:\KATETEST2.xyz"
```

```
vlc dvdsimple://D: :sout-transcode-vcodec=mp4v :sout- transcode-
vb=1024 :sout-transcode-acodec=mpga :sout-transcode-ab=192
:sout-standard-access=file :sout-standard-mux=ts :sout-standard-
dst="C:\KATETEST2.xyz"
```

```
vlc dvdsimple://D: :sout-transcode-ab=192 :sout-standard-
access=file :sout-standard-mux=ts :sout-standard-
dst="C:\KATETEST2.xyz" :sout-transcode-vcodec=mp4v :sout-
transcode-vb=1024 :sout-transcode-acodec=mpga
```

All of these commands produce the same incorrect effect. The stream is not output to a file, but displayed to the screen. No messages are displayed by VLC in the message log despite including the option for increased verbosity.

Files can be transcoded and displayed locally, though this has no real use and the output cannot be tested. The following command transcodes a DVD input stream and displays it to the screen.

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=mp4v,vb=1024,acodec=mpga,ab=192
}:duplicate{dst=display}
```

A file can also be transcoded and sent to the network. In the following example, a DVD input stream is transcoded and sent to a unicast UDP address.

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=h264,vb=1024,acodec=a52,ab=192}:duplicate{
dst=display,dst=std{access=udp,mux=ts,dst=192.168.2.4}}
```

To test the transcoded file, the stream was received on the client side, displayed, and saved to a file using the following command.

```
vlc udp:
:sout=#duplicate{dst=display,dst=std{access=file,mux=ts,dst="C:\
KateLovesVLC.xyz"}}
```

This produces a file that is recognized and played correctly by VLC, both audio and video. However, for an unknown reason, MediaInfo does not recognize the video stream. It is possible that this video codec (H264) is not recognized by MediaInfo. This codec is compatible with the container format based on the compatibility tables given in VideoLAN's Streaming Features website [5]. The container format and audio information are all as specified. This is the MediaInfo output for the file:

```
General #0
        Complete name        : G:\KateLovesVLC.xyz
        Format               : MPEG-1 Transport
        Format/Family        : MPEG-1
        File size            : 3.28 MiB
        PlayTime             : 1s 935ms
        Bit rate             : 14 Mbps

Audio #0
        Codec                : AC3
        PlayTime             : 1s 935ms
        Bit rate             : 192 Kbps
        Bit rate mode        : CBR
        Channel(s)           : 2 channels
        Sampling rate        : 48 KHz
```

```
ChannelPositions     : L R
```

This example demonstrates the compact syntax for the *transcode* module. The following command should transcode the DVD input stream into MP4.

```
vlc dvdsimple://D:
:sout=#transcode:std{access=file,acodec=mp4a,vcodec=mp4v,mux=mp4,
dst="C:\VLCTest\transcodeTest.abc"}
```

The syntax was modified slightly (*url* option was changed to *dst*) to test the *transcode* module's options.

```
vlc dvdsimple://D:
:sout=#transcode:std{access=file,acodec=mp4a,vcodec=mp4v,mux=mp4,
url="C:\VLCTest\transcodeTest.abc"}
```

Both commands did not produce any output and VLC gave this error message:

access_output_file error: cannot open `"F:\VLCTest\transcodeTest.abc"' (Invalid argument)
stream_out_standard error: no suitable sout access module for `file/mp4://"F:\VLCTest\transcodeTest.abc"'
stream_out_transcode error: cannot create chain
main error: stream chain failed for
`transcode:std{access=file,acodec=mp4a,vcodec=mp4v,mux=mp4,dst="F:\VLCTest\transcodeTest.abc"}'
main error: cannot start stream output instance, aborting

### 6.2.4 File to Network (UDP)

General syntax:

```
vlc input_stream --sout udp:ip_address

vlc input_stream --sout=std{access=udp,mux=mux_module,dst=ip_address}
```

Note that the 'std' module name is equivalent to 'standard' and can be used interchangeably.

To stream a DVD to a unicast address:

```
vlc dvd://D: --sout udp:192.168.2.4
```

A common option to add to this command is:

```
vlc dvd://D: --sout udp:192.168.2.4 --ttl 12
```

to set the TTL (Time To Live).
Both of these commands properly streamed output to the client computer where it was output for viewing to the screen.

This is the exact syntax given in VLC's documentation to stream to a UDP unicast address [4]:

```
vlc input_stream --sout
'#standard{access=udp,mux=ts,dst=ip_address}'
```

This was testing using the DVD drive as the input and streaming to the same client using the following command:

```
vlc dvdsimple://D: --sout
'#standard{access=udp,mux=avi,dst=192.168.2.4}'
```

However, this syntax is incorrect (the single quotes are unnecessary and an equals sign is missing between *sout* and *#standard*) and produces the following error messages.

stream_out_standard error: no access _and_ no muxer (fatal error)
main error: stream chain failed for `std{mux="",access="",dst="'#standard{access=udp,mux=avi,dst=192.168.2.5}'"}'
main error: cannot start stream output instance, aborting

### 6.2.5 File to Network (HTTP)

General syntax:

```
vlc input_stream --sout=std{access=http,mux=mux_module,dst=server_addr}
```

To stream a DVD using HTTP:

```
vlc dvdsimple://D:
--sout=#std{access=http,mux=ts,dst=192.168.2.10:8080}
```

The destination given here is actually the address of the server computer, where the client listens to. The client computer was able to properly access and display the stream.

This is the exact syntax given in VLC's documentation to stream to an HTTP address [4]:

```
vlc input_stream --sout
'#standard{access=http,mux=ogg,dst=server.example.org:8080}'
```

This was testing using the DVD drive as the input and streaming with the same server address as used in the correct commands above, using the following command:

```
vlc dvdsimple://D: --sout
'#standard{access=http,mux=ogg,dst=192.168.2.10:8080}'
```

However, this syntax is incorrect (the single quotes are unnecessary and an equals sign is missing between *sout* and *#standard*) and produces the following error messages.

stream_out_standard error: no access _and_ no muxer (fatal error)
main error: stream chain failed for `std{mux="",access="",dst="="}'
main error: cannot start stream output instance, aborting

```
stream_out_standard error: no access _and_ no muxer (fatal error)
main error: stream chain failed for `std{mux="",access="",dst="="}'
main error: cannot start stream output instance, aborting
```

To transcode the input and stream using HTTP, the following command was used:

```
vlc dvdsimple://D: --sout=
#transcode{vcodec=mp4v,acodec=mpga,vb=1024,ab=128}:
std{access=http,mux=asf,dst=192.168.2.10:8080}
```

The file was received on the client computer using the following command to save the stream to a file:

```
vlc http://192.168.2.5:8080 --sout=#std{acces
s=file,mux=asf,dst="C:\VLCKateTest.abc"}
```

The file was transcoded correctly, as demonstrated by the output of MediaInfo when compared to the specifications given for the transcoded file:

```
General #0
      Complete name          : C:\VLCKateTest.abc
      Format                 : Windows Media
      File size              : 3.96 MiB
      PlayTime               : 241h 56mn
      Bit rate               : 38 bps

Video #0
      Codec                  : MS MPEG-4 v3
      Codec/Info             : Microsoft MPEG-4 (Windows Media 7.0)
      Width                  : 720 pixels
      Height                 : 480 pixels
      Aspect ratio           : 1.500
      Standard               : NTSC

Audio #0
      Codec                  : MPEG-1/2 L3
      Codec/Family           : MPEG-1
      Codec/Info             : MPEG-1 or 2 layer 3
      Bit rate               : 128 Kbps
      Channel(s)             : 2 channels
      Sampling rate          : 48 KHz
```

This is the exact syntax given in VLC's documentation to transcode the input and stream using HTTP:

```
vlc input_stream --sout
'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:
standard{access=http,mux=ogg,dst=192.168.2.10:8080}'
```

This was testing using the DVD drive as the input and streaming with the same server address as used in the correct commands above, using the following command [4]:

```
vlc dvdsimple://D: --sout
'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:
standard{access=http,mux=ogg,dst=192.168.2.10:8080}'
```

This is the error message that was produced:

stream_out_standard error: no mux specified or found by extension
main error: stream chain failed for
`std{mux="",access="'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}",dst="standard{access=http,mux=ogg
,dst=192.168.2.10}'"}'
main error: cannot start stream output instance, aborting

### 6.2.6 Network to Screen (UDP)

Receiving a network stream on the client computer:

Unicast UDP address:

```
vlc udp:
```

The stream is received and displayed as expected.

Multicast UDP address:

```
vlc udp:@192.168.2.4
```

The stream is received and displayed as expected.

### 6.2.7 Network to Screen (HTTP)

Receiving a network stream on the client computer, using the address of the server computer:

```
vlc http://192.168.2.10:8080
```

The stream is received and displayed as expected.

### 6.2.8 Network to File (any network type)

To save a network stream being received to a file, add this option to the input command:

**`--sout=std{access=file,mux=`*`muxstring`*`,dst="`*`path`*`"}}`**

A shortcut for this command is also available, though it is recommended to use the above syntax:

**--sout file/*muxstring*:*path***

These options were tested with the following commands:

```
vlc http://192.168.2.10:8080 --sout=file/avi:C:\Test.avi

vlc http://192.168.2.10:8080 --sout=file/avi:C:\Test.xyz
```

Each of these saves the stream to the destination file. The file extension is not important.

As a more complete example, the following stream is transcoded and sent to the client computer:

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=mp4v,vb=512,acodec=mp3,ab=192}:duplicate{d
st=std{access=udp,mux=ts,dst=192.168.2.4}
```

The stream is received with the following command:

```
vlc udp: --
sout=#std{access=file,mux=ts,dst="C:\VLCTranscodeTest.xyz"}
```

The stream is saved to the file as expected and all of its attributes correspond correctly to those specified in the transcoding options. This is the MediaInfo output for the transcoded file and corresponds correctly to the specifications of the transcoded stream:

```
General #0
      Complete name        : C:\VLCTranscodeTest.xyz
      Format               : MPEG-4 Transport
      Format/Family        : MPEG-4
      File size            : 3.69 MiB
      PlayTime             : 59s 587ms
      Bit rate             : 519 Kbps

Video #0
      Codec                : MPEG-4 Video
      PlayTime             : 59s 585ms
      Bit rate             : 318 Kbps
      Width                : 720 pixels
      Height               : 480 pixels
      Aspect ratio         : 4/3
      Standard             : NTSC
      Resolution           : 8 bits
      Interlacement        : Progressive

Audio #0
      Codec                : MPEG-1 Audio layer 3
      Codec profile        : Joint stereo
      PlayTime             : 59s 587ms
      Bit rate             : 192 Kbps
      Bit rate mode        : CBR
```

```
Channel(s)           : 2 channels
Sampling rate        : 48 KHz
Resolution           : 16 bits
Video0 delay         : 18ms
```

### 6.2.9 Combinations

The following section gives various examples of more complex command chains, including sending a stream to two different output locations using different access methods and transcoding separate output streams differently depending on the destination.

Display the stream and send it to two unicast IP addresses:

```
vlc dvdsimple://D: --
sout=#duplicate{dst=display,dst=standard{access=udp,mux=ts,dst=1
92.168.2.5},dst=standard{access=udp,mux=ts,dst=192.168.2.4}}
```

The stream was received successfully by both client computers and displayed locally.

Send the original input stream to a unicast UDP address, then transcode the stream and send it to another unicast UDP address:

```
vlc dvdsimple://D: --
sout=#duplicate{dst=standard{access=udp,mux=ts,dst=192.168.2.4},
dst=transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:standard{ac
cess=udp,mux=ts,dst=192.168.2.5}}
```

The first client computer received the stream and was able to display it locally. The second client (receiving the transcoded stream) could not receive the stream at all. The access method, container format, and codecs all are compatible as described by the VideoLAN Streaming Features website [5] so it is unknown why the second client was unable to receive the stream. No error messages were given.

This is the output of MediaInfo from the file received by the first client computer:

```
General
        Complete name        : G:\twoComputers.xyz
        Format               : MPEG-2 Transport
        Format/Family        : MPEG-2
        File size            : 74.7 MiB
        PlayTime             : 2mn 14s
        Bit rate             : 4652 Kbps

Video
        Codec                : MPEG-2 Video
        PlayTime             : 2mn 14s
        Bit rate             : 7500 Kbps
        Bit rate mode        : CBR
        Width                : 720 pixels
        Height               : 480 pixels
```

```
        Aspect ratio         : 4/3
        Frame rate           : 29.970 fps
        Standard             : NTSC
        Interlacement        : Top Field First
        Bits/(Pixel*Frame)   : 0.724

Audio
        Codec                : AC3
        PlayTime             : 2mn 14s
        Bit rate             : 192 Kbps
        Bit rate mode        : CBR
        Channel(s)           : 2 channels
        Sampling rate        : 48 KHz
        Video0 delay         : -103ms
        ChannelPositions     : L R
```

The following command is almost exactly the same as the previous one, except that the transcoded stream is now being sent to the first client, and the original stream is being sent to the second client.

```
vlc dvdsimple://D: --
sout=#duplicate{dst=standard{access=udp,mux=ts,dst=192.168.2.4},
dst=”transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:standard{a
ccess=udp,mux=ts,dst=192.168.2.5}”}
```

The results are also similar. The client being sent the transcoded stream still could not receive it, even to display it, and the client sent the original stream was able to receive and display it. No error messages were produced.

This example should display the input stream locally, transcode it and send it to two unicast UDP addresses:

```
vlc dvdsimple://D: –-
sout=#duplicate{dst=display,dst=transcode{vcodec=mp4v,acodec=mpg
a,vb=800,ab=128}:duplicate{dst=standard{access=udp,mux=ts,dst=19
2.168.2.4},dst=standard{access=udp,mux=ts,dst=192.168.2.5}}
```

However, the stream could not be received by either client, even to display it to the screen. The stream was displayed locally as expected. No error messages were produced.

The syntax was adjusted slightly (quotations were inserted around the parameters of the *duplicate* module’s *dst* field) and re-entered:

```
vlc dvdsimple://D: --
sout=#duplicate{dst=display,dst="transcode{vcodec=mp4v,acodec=mp
ga,vb=800,ab=128}:duplicate{dst=standard{access=udp,mux=ts,url=1
92.168.2.4},dst=standard{access=udp,mux=ts,url=192.168.2.5}"}
```

This time, the stream did display locally on both clients and the server, but would not save to a file. No error messages were produced.

This next example is meant to display the stream, transcode it, and send it to a unicast UDP address:

```
vlc dvdsimple://D: –-
sout=#duplicate{dst=display,dst=transcode{vcodec=mp4v,acodec=mpg
a,vb=800,ab=128}:duplicate{dst=standard{access=udp,mux=ts,dst=19
2.168.2.4}}
```

The stream cannot be received by the client computer at all, even when using the Wizard to automate the process. No error messages were produced.

The syntax was adjusted slightly (the second duplicate chain is replaced with another *dst* option) and re-entered:

```
vlc dvdsimple://D: –-
sout=#duplicate{dst=display,dst=transcode{vcodec=mp4v,acodec=mpg
a,vb=800,ab=128,dst=standard{access=udp,mux=ts,dst=192.168.2.4}}
```

As with above, the stream cannot be received by the client computer at all.

In a final attempt, the syntax is adjusted to:

```
vlc dvdsimple://D: --
sout=#transcode{vcodec=mp4v,vb=512,acodec=mp3,ab=192}:duplicate{d
st=std{access=udp,mux=ts,dst=192.168.2.4}
```

and received by the client using the command:

```
vlc udp: -- sout=#std{access=file,mux=ts,dst="C:\VLCTest8.xyz"}
```

The output file has only audio, though the access method, container format, and codecs are all compatible according to the VideoLAN Streaming Features website [5].

### 6.2.10 Video Filters

This section gives examples of applying video filters to an input stream. Video filters are only applicable when displaying the input to the screen, as their effects cannot be streamed (only displayed locally).

For example, the following command displays the input stream locally and streams it to a client using UDP. Video filters are applied as a global option.

```
vlc dvdsimple://D: --filter=invert --
sout=#duplicate{dst=display,dst=std{access=udp,mux=ts,dst=192.16
8.2.4}}
```

As expected, the video is displayed on the server with its colors inverted. On the client computer, the video is displayed without any processing and the colors are normal. Filters

are only post processing the output and do not get streamed. If the client receives the stream from the network with the following command:

```
vlc udp: --filter=invert
```

the video will be displayed with inverted colors.

Filters must be cascaded (multiple filters applied at once) using the following syntax:

```
vlc input_stream --filter=filter1[:filter2]*
```

For example, to apply the motion blur effect and invert the video's colors, the following command is used:

```
vlc dvdsimple://D: --filter=motionblur:invert
```

These commands will not work, only the filter specified last will be applied:

```
vlc dvdsimple://D: --filter=motionblur --filter=invert
```

```
vlc dvdsimple://D: --filter=invert --filter=motionblur
```

In the first example, only the video's colors are inverted. In the second, only the motion blur effect is applied.

The *transform* filter is special because it takes parameters. To use this filter the following syntax is required:

```
vlc dvdsimple://D: --vout-filter=transform --transform-type hflip
```

This command properly flips the video horizontally, even though VLC produces this error message:

```
main error: option directx-hw-yuv does not exist
main error: option directx-device does not exist
```

Similarly, this command flips the video vertically:

```
vlc dvdsimple://D: --vout-filter=transform --transform-type vflip
```

The following syntax is also given as a method of using the *transform* filter:

```
vlc dvdsimple://D: --vout-filter=transform{type=hflip}
```

However, this simply generates and error message and causes no video to be output.

```
main error: no video filter module matched "transform{type=hflip}"
main error: no suitable vout module
main error: cannot delete object (303, (null)) with children
main error: failed to create video output
```

To combine the *transform* filters, the following command attempts to cascade the transform types. This command should flip the video vertically and then horizontally, as if it was rotated 180 degrees.

```
vlc dvdsimple://D: --vout-filter=transform --transform-type
vflip:hflip
```

However, the video appears as if it has been rotated by only 90 degrees. Also, changing the order of the transform filters, as in the following command:

```
vlc dvdsimple://D: --vout-filter=transform --transform-type
hflip:vflip
```

outputs the exact same video as the previous command, although intuitively it should also output the video as if it was rotated 180 degrees.

Any additional cascades (adding any number of additional vertical or horizontal transformation types) also results in the video being output in the same way, rotated by 90 degrees.

In the following chapter, the results of these tests will be examined and summarized to provide an overview of what syntax works and what does not.

# Chapter 7: Evaluation of Results

Despite the many examples given in VLC's documentation, all of these commands completely failed, not only to produce the desired output, but to produce any output at all. The only syntax format that actually works as expected is the one detailed in Chapter 5, section 5.3.2, discovered through the trial and error of the test cases. Following this format is the best way to guarantee consistent results. Actually, all of the syntax guidelines presented in Chapter 5 were modeled after the test cases that worked as expected and are the best summary of the results.

The problem with the command line examples as given exactly in VLC's documentation is that the command line interface is not actually operating system independent, despite claims to the contrary. Each operating system parses command line parameters differently, so while VLC may be receiving the exact same command from the operating system, what is input by the user will vary. For example, when setting the value of an option in Windows, often the argument must be placed in double quotation marks. This is necessary to stop Windows from recognizing certain special characters that the user did not intend (such as the backslashes used in a file name). In a Linux operating system, the entire stream output chain must be placed in single quotes for the command to be recognized. However, these single quotes are not necessary in Windows and change how the command is parsed. The commands taken directly from VLC's documentation are meant for a Linux operating system and will not work if used directly under Windows. This is likely the case for the other operating systems supported.

The most common transcoding and streaming problems arose from incompatibilities. Each method of streaming to the network is very limited in the container formats that it will support and the stream simply will not be sent if the format chosen is incompatible. Similarly when transcoding, if the audio or video codec is not compatible with the container format, that elementary stream will not be transcoded. It is safer to always double check these selections against the VideoLAN Streaming Features website. [5]

Most container formats have a standard file extension that they are associated with, but file extensions do not need to be consistent with their container format when saving a stream to a file. This was expressed as necessary in the documentation but is not required. [1] VLC will still recognize the file properly.

As there was very limited success in chaining the modules to perform more complex actions (as described in the Combinations section 6.2.9 of the previous chapter) it is a better idea, for now, to simply open multiple instances of VLC to perform multiple tasks simultaneously.

# Chapter 8: Conclusions and Future Work

In this thesis, I have examined the state of open source documentation and the problems that inherently arise from a large scale volunteer project. There is always the need for more, better documentation and open source projects need to look into recruiting capable writers or limiting volunteers to programmers who are willing to document their work. While this would likely discourage many programmers from contributing, it will lead to a product that is more reliable, maintainable, and useable. No matter how incredible a program may be, it is no good to anyone if it cannot be used easily and effectively.

This thesis tests the content of VLC's official documentation and combines much of the correct, complete, and up-to-date documentation together into a single document. New findings as to proper ways to use VLC and formulate proper syntax using its command line interface have also been added. However, this document is far from complete. These findings are not tested on previous versions of VLC to confirm whether the syntax has changed between recent versions. This would also likely be of benefit to users, especially those who find that a specific version performs better for their needs. The VideoLAN website provides a table of compatibilities between container formats, codecs, and output methods, but it would be worthwhile to thoroughly test these and confirm their correctness. This thesis is also limited to the command line interface, but VLC provides eight different specialty interfaces that have no documentation whatsoever, aside from a few paragraphs on the HTTP interface. Each interface should have its own documentation. There is plenty of room for future investigations as the VideoLAN project evolves.

# Bibliography

[1] A. de Lattre et al., “VideoLAN Streaming Howto,” 2005. [Online]. <http://www.videolan.org/doc/streaming-howto/en/index.html> [cited March 6, 2007]

[2] A. de Lattre et al., “VLC Play Howto,” 2006. [Online]. <http://www.videolan.org/doc/play-howto/en/play-howto-en.html> [cited March 6, 2007]

[3] H. Fallon et al., “VLC User Guide,” 2003. [Online]. <http://tldp.paracoda.com/REF/VLC-User-Guide/index.html> [cited March 6, 2007]

[4] VideoLAN Team, “VideoLAN FAQ,” 2006. [Online]. <http://www.videolan.org/doc/faq/en/videolan-faq-en.html> [cited March 6, 2007]

[5] VideoLAN Team, “VideoLAN Streaming Features List,” 2006. [Online]. <http://www.videolan.org/streaming-features.html> [cited March 6, 2007]

[6] VideoLAN Wiki, “Codec,” January 2007. [Online] <http://wiki.videolan.org/index.php/Codec> [cited March 6, 2007]

[7] VideoLAN Wiki, “Knowledge Base,” December 2006. [Online] <http://wiki.videolan.org/index.php/Knowledge_Base> [cited March 6, 2007]

# Appendix A: Annotated Bibliography

**VLC Streaming How-To**
A. de Lattre et al., "VideoLAN Streaming Howto," 2005. [Online]. <http://www.videolan.org/doc/streaming-howto/en/index.html> [cited March 6, 2007]

**VideoLAN Streaming Features List**
VideoLAN Team, "VideoLAN Streaming Features List," 2006. [Online]. <http://www.videolan.org/streaming-features.html> [cited March 6, 2007]

**VLC User Guide**
H. Fallon et al., "VLC User Guide," 2003. [Online]. <http://tldp.paracoda.com/REF/VLC-User-Guide/index.html> [cited March 6, 2007]

**VLC Updates**
VideoLAN Team, "News". [Online] <http://developers.videolan.org/vlc/NEWS> [cited March 6, 2007]

**VideoLAN Wiki Pages**
VideoLAN Wiki, "Documentation:Documentation," November 2006. [Online] <http://wiki.videolan.org/Documentation:Documentation> [cited March 6, 2007]

**VLC Knowledge Base (from the VideoLAN Wiki)**
VideoLAN Wiki, "Knowledge Base," December 2006. [Online] <http://wiki.videolan.org/index.php/Knowledge_Base> [cited March 6, 2007]

**VLC Command Line Help (from the VideoLAN Wiki)**
VideoLAN Wiki, "VLC Command Line Help," January 2007. [Online] <http://wiki.videolan.org/index.php/VLC_command-line_help> [cited March 6, 2007]

**VLC Play How-To**
A. de Lattre et al., "VLC Play Howto," 2006. [Online]. <http://www.videolan.org/doc/play-howto/en/play-howto-en.html> [cited March 6, 2007]

**VLC API Documentation**
C. Massiot et al., "VLC media player API Documentation," 2001. [Online]. <http://www.videolan.org/developers/vlc/doc/developer/html/> [cited March 6, 2007]

**VLC Frequently Asked Questions**
VideoLAN Team, "VideoLAN FAQ," 2006. [Online].

<http://www.videolan.org/doc/faq/en/videolan-faq-en.html> [cited March 6, 2007]

**VideoLAN Play How-To/Advanced Use of VLC (from VideoLAN Wiki)**

VideoLAN Wiki, "Documentation:Play HowTo/Advanced Use of VLC," February 2007. [Online]. <http://wiki.videolan.org/Documentation:Play_HowTo/Advanced_Use_of_VLC> [cited March 6, 2007]

**VLC Doxygen Documentation**

Doxygen, "VLC Documentation," March 6, 2007. [Online]. <http://www.videolan.org/developers/vlc/doc/doxygen/html/> [cited March 6, 2007]

**VLC Linux How-To Guide**

A. de Lattre et al., "VideoLAN HOWTO," 2003. [Online] <http://www.linux.org/docs/ldp/howto/VideoLAN-HOWTO/index.html> [cited March 6, 2007]

# Appendix B: Glossary of Terms

| | |
|---|---|
| Codec | A program capable of encoding or decoding a data stream (specifically a video or audio stream). [7] |
| Container format | A file format that can contain multiple types of elementary streams (usually one for audio and one for video). These streams have already been encoded. Unfortunately, container formats are only compatible with certain codecs. [1] |
| Decoder | Program that uses mathematical processing to decompress the elementary stream. [3] |
| Demultiplexer | Reads the container format and separates the data streams contained within into separate files. Audio, video, and subtitle elementary streams are separated so they can be decoded.[3] |
| Elementary stream | A stream containing only one type of data (in this case, either audio, video, or subtitle) |
| Encoder | A compression algorithm used to reduce the size of an elementary stream. [3] |
| HTTP | Hypertext Transfer Protocol. Used to convey information over the world wide web. [7] |
| MMS | Stream using the Microsoft Media Server (MMS) protocol to transfer unicast data. MMS can be transported via UDP or TCP. [7] |
| Multicast | The delivery of information to a group of destinations simultaneously using the most efficient strategy to deliver the messages over each link of the network only once, creating copies only when the links to the destinations split. [7] |
| Multiplexer | Used to combine the elementary data streams (audio and video) into one file, a container format [1] |
| RTP | Real Time Transport Protocol. A standardized packet format for delivering audio and video over the Internet. Applications using RTP are less sensitive to packet loss, but typically very sensitive to delays. [7] This protocol is used for unreliable delivery of Real Time data that is layered on top of UDP (and thus can make use of multicast). It is used mainly for streaming audio and video often controlled by an RTSP session. [7] |
| RTSP | Real Time Streaming Protocol. A client-server multimedia presentation control protocol which allows a client to remotely control a streaming media server, allowing time-based access to files on a server. [7] |
| SAP | Session Announcement Protocol. A protocol for broadcasting multicast session information. [7] |
| TTL | Time To Live, the number of routers your stream will be able to cross. [1] |
| UDP | User Datagram Protocol, known as the “send and pray” protocol. Data is passed over the network in fixed-size packets but does not |

| | |
|---|---|
| | guarantee that all packets will reach the destination, or that they will reach it in the right order. Because it will not resend packets, it is very fast and efficient, but also unreliable. However, for streaming video this is not usually an issue, making UDP very suitable for this type of transmission. [7] |
| Unicast | Sending of information packets to a single destination over a network. |

# Appendix C: Test File Specifications

This appendix includes the MediaInfo output for each of the original files used in the test cases. Some of the irrelevant information from the output has been removed as it is unnecessary for this project.

**General**

| | |
|---|---|
| Complete name | : C:\VLCTest\Sliders4.avi |
| Format | : AVI |
| Format/Family | : RIFF |
| File size | : 120 MiB |
| PlayTime | : 43mn 47s |
| Bit rate | : 377 Kbps |

**Video**

| | |
|---|---|
| Codec | : XviD |
| Codec/Family | : MPEG-4 |
| Codec/Info | : XviD project |
| PlayTime | : 43mn 47s |
| Bit rate | : 316 Kbps |
| Width | : 576 pixels |
| Height | : 432 pixels |
| Aspect ratio | : 4/3 |
| Frame rate | : 23.976 fps |
| Resolution | : 8 bits |
| Chroma | : 4:2:0 |
| Interlacement | : Progressive |
| Bits/(Pixel*Frame) | : 0.053 |

**Audio**

| | |
|---|---|
| Codec | : MPEG-2 Audio layer 3 |
| Codec profile | : Joint stereo |
| PlayTime | : 43mn 47s |
| Bit rate | : 48 Kbps |
| Bit rate mode | : CBR |
| Channel(s) | : 2 channels |
| Sampling rate | : 24 KHz |
| Resolution | : 16 bits |

**DVD Drive (Todd McFarlane's Spawn, animated)**
**General**

| | |
|---|---|
| Complete name | : D: |
| Format | : MPEG-2 Program |
| Format/Family | : MPEG-2 |

| | |
|---|---|
| File size | : 80.5 MiB |
| PlayTime | : 1mn 55s |
| Bit rate | : 5826 Kbps |

**Video**

| | |
|---|---|
| Codec | : MPEG-2 Video |
| PlayTime | : 1mn 53s |
| Bit rate | : 7500 Kbps |
| Bit rate mode | : CBR |
| Width | : 720 pixels |
| Height | : 480 pixels |
| Aspect ratio | : 4/3 |
| Frame rate | : 29.970 fps |
| Standard | : NTSC |
| Chroma | : 4:2:0 |
| Interlacement | : Top Field First |
| Bits/(Pixel*Frame) | : 0.724 |

**Audio**

| | |
|---|---|
| Codec | : AC3 |
| PlayTime | : 1mn 55s |
| Bit rate | : 384 Kbps |
| Bit rate mode | : CBR |
| Channel(s) | : 6 channels |
| Sampling rate | : 48 KHz |
| Video0 delay | : -88ms |
| ChannelPositions | : Front: L C R, Rear: L R, Subwoofer |

# Appendix D: VLC Keywords

The following lists are the VLC keywords that are used as options in the transcoding and standard modules. These lists are taken from the VideoLAN Wiki pages [6]. This is not official VideoLAN documentation and all of the following commands have not been properly tested, so there is no guarantee that they are all correct. Also, the keywords do not seem to be case sensitive.

## Demultiplexers

A demultiplexer can be forced in priority by adding the following command line argument:

```
--demux demux_module
```

where demux_module is one of the keywords listed below.

| Name | Description |
|---|---|
| avi | This module allows VLC to read `.avi` files and is always enabled. |
| asf | This module allows VLC to read `.asf` files and is always enabled. |
| aac | This module allows VLC to read AAC files and is always enabled. |
| ogg | This module allows VLC to read .ogg files and is enabled by default. |
| rawdv | This module allows VLC to read DV files and is always enabled. |
| dvbpsi | This module allows VLC to read streams from a satellite card and is enabled by default. |
| mp4 | This module allows VLC to read .mp4 files and is always enabled. |
| mkv | This module allows VLC to read files that use the Matroska free format and is enabled by default. |
| ps | This module allows VLC to read MPEG2 Program Stream files and is always enabled. |
| ts | This module allows VLC to read MPEG2 Transport Stream files and is always enabled. |
| id3, m3u | This module allows VLC to read M3U, B4S, PLS, and ASX playlists, and ID3 tags and is always enabled. |

[3]

## Video Codecs

A codec can be forced in priority by adding the following command line argument:

```
--codec codec_module
```

where codec_module is one of the keywords listed below.

Use the "Name" column in your vcodec=<string> commands when transcoding a file.

| Name | Description |
|---|---|
| mp1v | MPEG-1 Video - recommended for portability |
| mp2v | MPEG-2 Video - used in DVDs |
| mp4v | MPEG-4 Video |
| SVQ1 | Sorenson Video v1 |
| SVQ3 | Sorenson Video v3 |
| WMV1 | Windows Media Video v1 |
| WMV2 | Windows Media Video v2 |
| WMV3 | Windows Media Video v3, also called Windows Media 9 (unsupported) |
| DVSD | Digital Video |
| MJPG | MJPEG |
| H263 | H263 |
| h264 | H264 |
| theo | Theora |
| IV20 | Indeo Video |
| IV40 | Indeo Video version 4 or later (unsupported) |
| RV10 | Real Media Video |
| cvid | Cinepak |
| VP31 | On2 VP |
| FLV1 | Flash Video |
| CYUV | Creative YUV |
| HFYU | Huffman YUV |
| MSVC | Microsoft Video v1 |
| MRLE | Microsoft RLE Video |
| AASC | Autodesc RLE Video |
| FLIC | FLIC video |
| QPEG | QPEG Video |

[6]

# Audio Codecs

A codec can be forced in priority by adding the following command line argument:

```
--codec codec_module
```

where codec_module is one of the keywords listed below.

Use the "Name" column in your acodec=<string> commands when transcoding a file.

| Name | Description |
|---|---|
| mpga | MPEG audio (recommended for portability) |
| mp3 | MPEG Layer 3 audio |
| mp4a | MP4 audio |
| a52 | Dolby Digital (A52 or AC3) |
| vorb | Vorbis |
| spx | Speex |
| flac or fl32 | FLAC |

[6]

# Muxers

Use the "Name" column in you mux=<string> commands when streaming a file.

| Name | Description |
|---|---|
| mpeg1 | MPEG-1 multiplexing - recommended for portability. This muxer should be used instead of ps with MPEG 1 video streams, when saved to a file or streamed over HTTP. Supported codecs are MPEG 1 and MPEG audio. |
| ts | MPEG Transport Stream, primarily used for streaming MPEG. Also used in DVDs. This the standard muxer used to stream MPEG 2. This muxer can be used with any *access* method. Supported codecs are MPEG 1/2/4, MJPEG, H263, H264, I263, WMV 1/2 and theora for video, MPEG audio, AAC and a52 for the audio stream. [1] |
| ps | MPEG Program Stream. This the standard muxer for MPEG 2 files(.mpg). It can be used with the file and http output methods. Supported codecs are MPEG 1/2 and MJPEG for video, MPEG audio and a52 for audio streams. The only available item option is *dst-delay=<delay in ms>*. It allows the user to delay PTS (Presentation Time Stamps) from the DTS (Decoding Time Stamp) from the given time. |
| mp4 | MPEG-4 mux format, used only for MPEG-4 video and MPEG audio. |
| avi | The Microsoft AVI muxer. This is very common encapsulation format for MPEG 4 files. The only supported output method is file. Supported codecs are MPEG 1/2/4, H263, H264 and I263 for video, MPEG audio and a52 for audio streams. There are no item options for this muxer. [1] |
| asf | The Microsoft ASF muxer. This is the standard muxer used for streaming by Microsoft's software. Is also used as container for WMA audio files. This muxer can be used with the file and HTTP output methods. Supported codecs are MPEG 4, MJPEG, WMV 1/2 for video, MPEG audio, a52 for audio streams. [1] |
| asfh | This is a special version of the ASF muxer, that should be used for MMSH |

| | |
|---|---|
| | streaming. MMSH is the only supported output method. Supported codecs are the same as for ASF. [1] |
| dummy | dummy output, can be used in creation of MP3 files. |
| ogg | The ogg muxer. This is the muxer from the Xiph project. It can be used with the HTTP and file output methods. Supported codecs are MPEG 1/2/4, MJPEG WMV 1/2 and Theora, audio streams can be vorbis, flac, speex, a52 or MPEG audio. There are no item options for this muxer. [1] |
| mpjpeg | The multipart jpeg muxer. This encapsulation format is mostly used on surveillance video cameras with an integrated web-server. Such streams are usually embedded in web-pages and seen with standard Internet browsers, as they are seen as a succession of jpeg images. The only supported output method is HTTP. The only usable codec is MJPEG. No sound track can be muxed in such streams. No item options are available for this muxer. [1] |

[6]

# Video Filters

These filters allow modifications to the rendered image. Note that these filters only apply to the on screen display and thus cannot be streamed. They just format how your video is displayed once VLC receives it [2]. In order for filters to take effect on the client machine, they would have to be set on the client machine. All of these filters are always enabled, so they do not require any special configurations to be used.

Filters are applied by adding one of the following equivalent command line arguments:

```
--filter=filter_name1[:filter_name2]*

--vout-filter=filter_name1[:filter_name2]*
```

where filter_nameX is one of the filter keywords listed below. This is a complete list of all filters as described in the VLC User Guide [3]. The descriptions have been elaborated on where possible.

| **Name** | **Description** |
|---|---|
| deinterlace | This filter deinterlaces video. It is useful with streams coming from a digital satellite channel or digital terrestrial television channels. |
| wall | This filter allows you to have the video cut in pieces in several windows, which you can order as you wish. It can be used to generate image walls with several sources. |
| distort | This filter adds a distortion effect to the video. |
| transform | This filter allows you to rotate the video in several ways, as well as flip the video horizontally or vertically. This filter requires parameters (either the angle of rotation or the type of flipping). |
| invert | This filter inverts all colors. |
| adjust | This filter allows you to set image contrast, hue, saturation and |

| | |
|---|---|
| | brightness. |
| clone | This filter allows you to duplicate the image and display it in more than one window. |
| crop | This filter allows you to crop parts of the image. |
| motionblur | This filter adds a "motion blur" effect to the image. I tried applying this filter to all of the test input files and did not notice any difference from when I had not applied any filter, though the user interface informed me that the filter was in effect. |

[3]

# Appendix E: Command Line Options

This is a small reference of common command line options and parameters that are likely to be the most useful. Similar options are grouped under common headings and, where possible, the explanation given by the original help file has been elaborated on.

To get the full command line help document, use the following command:

**vlc --longhelp --advanced --help-verbose**

The full help listing will be dumped to a file in your installation folder.

## Video

```
-V, --vout=<string>
       Video output module
       This is the video output method used by VLC. The default
       behavior is to automatically select the best method available.
--video-filter=<string>
       Video filter module
       This adds post-processing filters to enhance the picture
       quality, for instance deinterlacing, or distort the video.
       <string> can be substituted for any of the options described in
       the Video Filters section of Appendix D
--vout-filter=<string>
       Video filter module alternative syntax
--filter=<string>
       Video filter module alternative syntax
```

## Miscellaneous

```
--server-port=<integer>
       UDP port
       This is the default port used for UDP streams. Default is 1234.
--mtu=<integer>
       MTU of the network interface
       This is the maximum packet size that can be transmitted over the
       network interface. On Ethernet it is usually 1500 bytes.
--clock-synchro={-1 (Default), 0 (Disable), 1 (Enable)}
       Clock synchronization
       It is possible to disable the input clock synchronization for
       real-time sources. Use this if you experience jerky playback of
       network streams.
--network-synchronisation, --no-network-synchronisation
       Network synchronization (default disabled)
       This allows you to remotely synchronize clocks for server and
       client. The detailed settings are available in Advanced /
       Network Sync. (Command is valid as shown even though
       “synchronization” is misspelled)
```

```
--plugin-path=<string>
       Modules search path
       Additional path for VLC to look for its modules.
--high-priority, --no-high-priority
       Increase the priority of the process (default disabled)
       Increasing the priority of the process will very likely improve
       your playing experience as it allows VLC not to be disturbed by
       other applications that could otherwise take too much processor
       time. However be advised that in certain circumstances (bugs)
       VLC could take all the processor time and render the whole
       system unresponsive which might require a reboot of your
       machine.
-v, --verbose=<integer>
       Verbosity (0,1,2)
       This is the verbosity level (0=only errors and standard
       messages, 1=warnings, 2=debug).
--file-logging, --no-file-logging
       Log to file (default disabled)
       Log all VLC messages to a text file.
--stats, --no-stats
       Collect statistics (default enabled)
       Collect miscellaneous statistics.
```

# CPU

```
--fpu, --no-fpu
       Enable FPU support (default enabled)
       If your processor has a floating point calculation unit, VLC can
       take advantage of it.
--mmx, --no-mmx
       Enable CPU MMX support (default enabled)
       If your processor supports the MMX instructions set, VLC can
       take advantage of them.
--3dn, --no-3dn
       Enable CPU 3D Now! support (default enabled)
       If your processor supports the 3D Now! instructions set, VLC can
       take advantage of them.
--mmxext, --no-mmxext
       Enable CPU MMX EXT support (default enabled)
       If your processor supports the MMX EXT instructions set, VLC can
       take advantage of them.
--sse, --no-sse
       Enable CPU SSE support (default enabled)
       If your processor supports the SSE instructions set, VLC can
       take advantage of them.
--sse2, --no-sse2
       Enable CPU SSE2 support (default enabled)
       If your processor supports the SSE2 instructions set, VLC can
       take advantage of them.
```

# Decoders

```
--codec=<string>
       Preferred decoders list
       List of codecs that VLC will use in priority. For instance,
       'dummy,a52' will try the dummy and a52 codecs before trying the
       other ones. Only advanced users should alter this option as it
       can break playback of all your streams.
--encoder=<string>
       Preferred encoders list
       This allows you to select a list of encoders that VLC will use
       in priority.
```

# Input

```
--access=<string>
       Access module
       This allows you to force an access module. You can use it if the
       correct access is not automatically detected. You should not set
       this as a global option unless you really know what you are
       doing.
--access-filter=<string>
       Access filter module
       Access filters are used to modify the stream that is being read.
       This is used for instance for timeshifting.
--demux=<string>
       Demux module
       Demultiplexers are used to separate the "elementary" streams
       (like audio and video streams). You can use it if the correct
       demuxer is not automatically detected. You should not set this
       as a global option unless you really know what you are doing.
```

# Stream output

```
--sout=<string>
       Default stream output chain
       You can enter here a default stream output chain. Refer to the
       documentation to learn how to build such chains. Warning: this
       chain will be enabled for all streams.
--sout-keep, --no-sout-keep
       Keep stream output open (default disabled)
       This allows you to keep a unique stream output instance across
       multiple playlist item (automatically insert the gather stream
       output if not specified).
--sout-all, --no-sout-all
       Enable streaming of all ES (default disabled)
       Stream all elementary streams (video, audio and subtitles)
--sout-audio, --no-sout-audio
       Enable audio stream output (default enabled)
       Choose whether the audio stream should be redirected to the
       stream output facility when this last one is enabled.
--sout-video, --no-sout-video
       Enable video stream output (default enabled)
```

```
        Choose whether the video stream should be redirected to the
        stream output facility when this last one is enabled.
```

# File input

```
--file-caching=<integer>
        Caching value in ms
        Caching value for files. This value should be set in
        milliseconds.
```

# FTP input

```
--ftp-caching=<integer>
        Caching value in ms
        Caching value for FTP streams. This value should be set in
        milliseconds.
--ftp-user=<string>
        FTP user name
        User name that will be used for the connection.
--ftp-pwd=<string>
        FTP password
        Password that will be used for the connection.
--ftp-account=<string>
        FTP account
        Account that will be used for the connection.
```

# HTTP input

```
--http-proxy=<string>
        HTTP proxy
        HTTP proxy to be used. It must be of the form
        http://[user[:pass]@]myproxy.mydomain:myport/
        If empty, the http_proxy environment variable will be tried.
--http-caching=<integer>
        Caching value in ms
        Caching value for HTTP streams. This value should be set in
        milliseconds.
--http-user-agent=<string>
        HTTP user agent
        User agent that will be used for the connection.
--http-reconnect, --no-http-reconnect
        Auto re-connect (default disabled)
        Automatically try to reconnect to the stream in case of a sudden
        disconnect.
```

# TCP input

```
--tcp-caching=<integer>
```

```
        Caching value in ms
        Caching value for TCP streams. This value should be set in
        milliseconds.
```

# UDP/RTP input

```
--udp-caching=<integer>
        Caching value in ms
        Caching value for UDP streams. This value should be set in
        milliseconds.
--rtp-late=<integer>
        RTP reordering timeout in ms
        VLC reorders RTP packets. The input will wait for late packets
        at most the time specified here (in milliseconds).
--udp-auto-mtu, --no-udp-auto-mtu
        Autodetection of MTU (default enabled)
        Automatically detect the line's MTU. This will increase the size
        if truncated packets are found.
```

# AVI demuxer

```
--avi-interleaved, --no-avi-interleaved
        Force interleaved method (default disabled)
--avi-index={0 (Ask), 1 (Always fix), 2 (Never fix)}
        Force index creation
        Recreate an index for the AVI file. Use this if your AVI file is
        damaged or incomplete (not seekable).
```

# Clone video filter

```
--clone-count=<integer>
        Number of clones
        Number of video windows in which to clone the video.
--clone-vout-list=<string>
        Video output modules
        You can use specific video output modules for the clones. Use a
        comma-separated list of modules.
```

# FFmpeg audio/video decoder/encoder ((MS)MPEG4,SVQ1,H263,WMV,WMA)

```
--ffmpeg-dr, --no-ffmpeg-dr
        Direct rendering (default enabled)
--ffmpeg-hurry-up, --no-ffmpeg-hurry-up
        Hurry up (default disabled)
```

```
        The decoder can partially decode or skip frame(s) when there is
        not enough time. It's useful with low CPU power but it can
        produce distorted pictures.
--ffmpeg-lowres=<integer>
        Low resolution decoding
        Only decode a low resolution version of the video. This requires
        less processing power definition streams.
--ffmpeg-pp-q=<integer>
        Post processing quality
        Quality of post processing. Valid range is 0 to 6. Higher levels
        require considerable more CPU power, but produce better looking
        pictures.
--sout-ffmpeg-hq={rd,bits,simple}
        Quality level
        Quality level for the encoding of motions vectors (this can slow
        down the encoding very much).
--sout-ffmpeg-hurry-up, --no-sout-ffmpeg-hurry-up
        Hurry up (default disabled)
        The encoder can make on-the-fly quality tradeoffs if your CPU
        can't keep up with the encoding rate. It will disable trellis
        quantization, then the rate distortion of motion vectors (hq),
        and raise the noise reduction threshold to ease the encoder's
        task.
```

# RTP/RTSP/SDP demuxer (using Live555)

```
--rtsp-tcp, --no-rtsp-tcp
        Use RTP over RTSP (TCP) (default disabled)
--rtp-client-port=<integer>
        Client port
        Port to use for the RTP source of the session
--rtsp-caching=<integer>
        Caching value (ms)
        Allows you to modify the default caching value for RTSP streams.
        This value should be set in millisecond units.
--rtsp-user=<string>
        RTSP user name
        Allows you to modify the user name that will be used for
        authenticating the connection.
--rtsp-pwd=<string>
        RTSP password
        Allows you to modify the password that will be used for the
        connection.
```

# ASF muxer

```
--sout-asf-packet-size=<integer>
        Packet Size
        ASF packet size -- default is 4096 bytes
```

# MP4/MOV muxer

```
--sout-mp4-faststart, --no-sout-mp4-faststart
        Create "Fast Start" files (default enabled)
        "Fast Start" files are optimized for downloads and allow the
        user to start previewing the file while it is downloading.
```

# MPEG Transport Stream demuxer

```
--ts-out=<string>
        Fast udp streaming
        Sends TS to specific ip:port by udp (you must know what you are
        doing).
--ts-out-mtu=<integer>
        MTU
        The size of the largest packet that a network protocol can
        transmit for out mode
--ts-csa-pkt=<integer>
        Packet size in bytes to decrypt
        Specify the size of the TS packet to decrypt. The decryption
        routines subtract the TS-header from the value before
        decrypting.
```

# Appendix F: CS4997 Summary Sheet

UNIVERSITY OF NEW BRUNSWICK
FACULTY OF COMPUTER SCIENCE
Fall 2006

STUDENT NAME: Kate Kinnear

STUDENT SIGNATURE: _____

STUDENT ID #: 3165595

E-MAIL: r5f67 @ unb.ca

PHONE: ( 506 ) 474 - 0734

THESIS TITLE: Investigation of the Functionality of VLC

SUPERVISOR: John DeDourek
(please print name)

DATE SUBMITTED: March 29, 2007

| PHASE TITLE | ESTIMATE | | ACTUAL | |
|---|---|---|---|---|
| | PERSON-HOURS | COMPLETION DATE | PERSON-HOURS | COMPLETION DATE |
| Annotated Bibliography | 10 | Dec 18, 2006 | 8 | Dec 9, 2006 |
| Choose Functions | 15 | Jan 12, 2007 | 22 | Jan 19, 2007 |
| Investigation | 60 | Mar 2, 2007 | 48 | Mar 3, 2007 |
| Test Case | 15 | Mar 2, 2007 | 20 | Mar 3, 2007 |
| Prediction | 10 | Mar 2, 2007 | 6 | Mar 3, 2007 |
| Testing | 5 | Mar 2, 2007 | 23 | Mar 3, 2007 |
| Analysis of Results | 5 | Mar 2, 2007 | 8 | Mar 3, 2007 |
| Thesis draft | 15 | Mar 8, 2007 | 24 | Mar 6, 2007 |
| Thesis | 5 | Mar 29, 2007 | 2 | Mar 28, 2007 |
| Presentation | 5 | Mar 21, 2007 | 6 | Mar 21, 2007 |
| | Total: 145 | | Total: 167 | |