

LINUX VOICE MASTERCLASS

Essential Linux tools explained – this month, the VLC media player and a trio of command line media tools

VLC MEDIA PLAYER – THE ONE THAT DOES IT ALL

Play media, transcode files, stream live media and more with this ancient application.

JOHN LANE

The VLC Media Player is a free and open-source multimedia player. There are a number of Linux media players to choose from but this one can play just about anything and will often succeed where other players fail.

But it is much more than just a media player. It was originally a pair of applications: the VideoLAN Client called **vlc** and the VideoLAN Server called **vlsv**. The client and server functionality are now both contained in a single application – the VLC Media Player, which everyone now refers to as VLC.

You can use VLC to encode videos in a similar way to the command-line tools that we looked at earlier and you can use it as a streaming server. But you'll probably want to start by using it as a player.

Playlist

Central to VLC is its playlist, which is more than your typical playlist – you can create, load and save your own playlists, but what we have here is more of a gateway to content, both on your local network and

on the internet. You can directly access media on your local machine, either its hard drive or DVD. If you plug in a media player that uses the Media Transfer Protocol (MTP) you can access its contents directly.

If you have any DLNA (Digital Living Network Alliance – we know, it sounds horrible) devices such as a PVR machine on your network, then you can also access their content through the playlist. DLNA is a media sharing standard offered on some internet-ready audio-visual devices like televisions and PVRs, and it is a simplified subset of something called Universal Plug And Play, which VLC also supports. Opening the playlist (Ctrl+P) and selecting Universal Plug'n'Play will query the devices on the network (this may take some time) and then offer a list of content. Click on an item to play it. Other network services can be accessed in a similar way – VLC supports SAP, the Session Announcement Protocol, which gives access to published multicast services and Apple's Bonjour protocol.

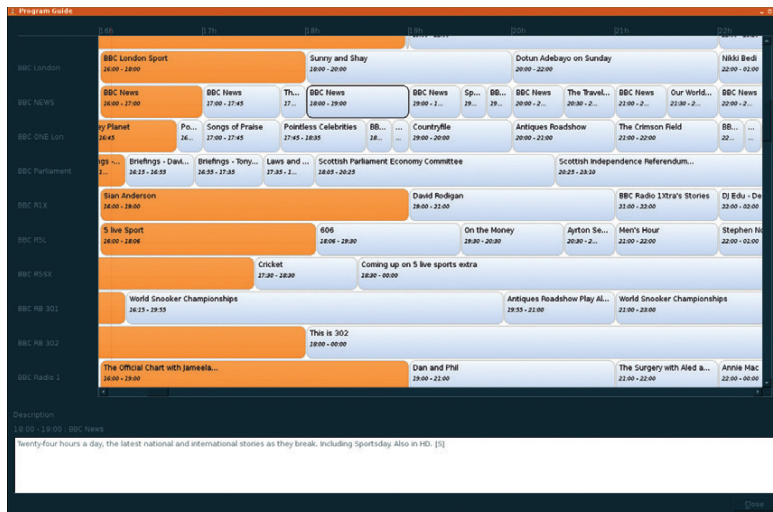
You can even directly access Freeview digital television broadcasts if you have a working TV card and know your local transmitter's frequencies. Go to Media > Open Capture Device and select the TV – Digital capture mode. Enter the relevant frequency and press Play. Programme guide information is collected and you can view this with Tools > Program Guide.

Internet media

Beyond your local network and TV antenna, the internet is a rich source of media, and VLC gives easy access to internet radio sources like Icecast and Jamendo through the Internet choices on the Playlist screen. You can also add your own media favourites, such as your favourite Linux podcasts.

To add a podcast, go to View > Playlist, then click on Podcasts; the Plus sign opens a dialog where you can enter the URL for the podcast (for example, www.linuxvoice.com/podcast_opus.rss), giving you

VLC's programme guide lets you see what's on TV.



direct access to all episodes of the podcast that the feed provides.

The default user interface fits in with your desktop environment, but VLC is skinnable, meaning that you can change its visual appearance. Many skins are offered at the VLC website, www.videolan.org/vlc/skins.php, and you can even download them all at once like so:

```
mkdir -p ~/.local/share/vlc/skins2
```

```
curl http://www1.videolan.org/vlc/skins2/vlc-skins.zip | bsdtar -C ~/.local/share/vlc/skins2 -xvf-
```

You need to go to the Preferences page (Control+P) within VLC and set the Look And Feel to “Use Custom Skin” and then exit and re-start. You can then choose a skin by right-clicking Interface > Select Skin.

You can also choose a skin when starting VLC using the `--skins2-last` command-line option. You need the full path to the skin file:

```
vlc --skins2-last=~/.local/share/vlc/skins2/neon.vlt
```

Be aware, however, that using skins can hide some of VLC’s functions.

Broadcast yourself

Once you’re happy that you can play pretty much anything you want, you can move on to streaming, because VLC will stream anything that it can play. You can select anything from your playlist and stream it – just use the drop-down menu on the Play button to change it to Stream.

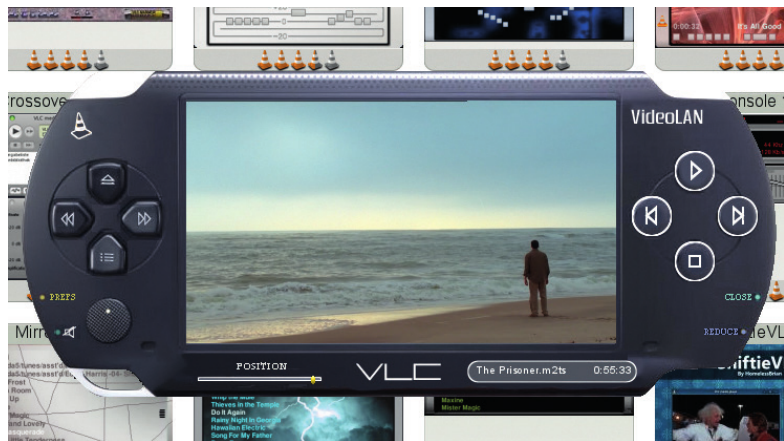
You’ll be walked through a set of screens where you specify what and how to stream – the protocol to use and any transcoding to perform. The first screen defines what will be streamed, and you can fill this in yourself, but selecting from the playlist will fill it in for you. The second screen enables you to define streaming destinations. These can be a file or one of several network profiles including HTTP and RTP (you can have more than one if you need to). You can also choose whether the streamed content is displayed locally so you can see what’s being streamed.

The fourth screen is where you can enable transcoding, and there are several configurations to choose from. If none of those suit, you can create your own. Once the stream starts, you can view it from any player that can access it (another copy of VLC is an obvious choice here).

Streaming to a file enables you to record a live stream to your local machine. You can also save remote content locally, which is a good way to take a copy of recordings from your PVR onto your computer. You can transcode while doing this if you want the copy in a different format to the original.

Your preference

We already mentioned how you can change the look and feel with custom skins. You can also change the behaviour with Preferences (Control+P). There are many options available here to change VLC’s behaviour, but it’s unlikely that you will need to change many of them. If you want to back up your



Change VLC’s look and feel with a custom skin.

preferences, they’re stored in a file:

```
~/.config/vlc/vlcrc
```

One of the preferences controls the video output. It’s normally set to automatic but, for fun, try setting it to Color ASCII art video output. You need to re-start VLC after changing preferences but, once you’ve done this, any video played will be rendered using colour ASCII art. This isn’t much use unless you’ve only got a text terminal, but it’s fun nonetheless and just goes to show that VLC has many tricks up its sleeve.

You can use the command-line version of VLC to view a movie in any terminal window. Now you don’t even need a graphical desktop to watch things! (press Alt+F4 to quit.)

```
cvlc -V caca my_movie.mp4
```

No GUI needed

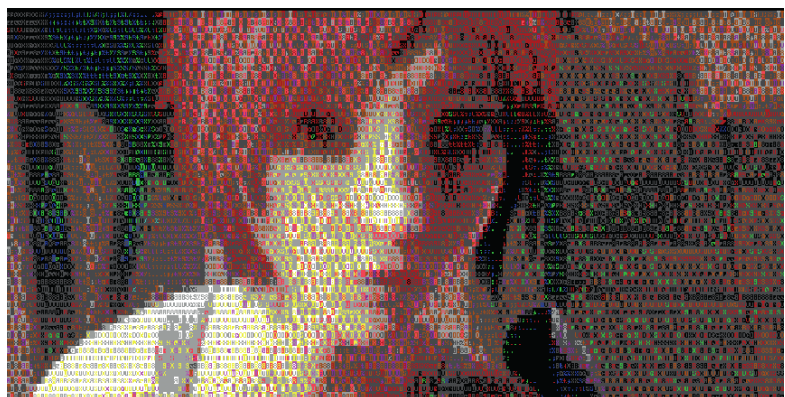
If you’re using VLC for downloading or streaming or you have another reason why you don’t need the full GUI experience then

there is a command-line version called CVLC, which offers the same functionality but is driven through command-line options. This is its server

heritage showing through, and CVLC enables you to use VLC for non-GUI applications such as setting up a streaming server or as a transcoding tool.

“Streaming to a file with VLC enables you to record a live stream to your local machine.”

With VLC’s ASCII art, you can watch a movie on a terminal.



FFMPEG AND MENCODER: COMMAND LINE VIDEO

Use your command-line prowess to make media files for any device.

As with most things Linux, there is a perplexing number of media tools available. Many of these are merely wrappers around a small number of utilities that perform the heavy lifting and the most common ones include Mencoder and FFmpeg, which we're going to look at here.

Mencoder comes from the same stable as the MPlayer media player, and it enables audio and video files to be decoded, filtered and re-encoded. As it's built from the same codebase as MPlayer, if MPlayer can play it, Mencoder can decode it.

FFmpeg is a collection of tools and libraries for encoding, decoding and streaming audio and video. It includes the libavcodec library, which is widely used by other packages including Mencoder. FFmpeg claims to be able to work with pretty much any multimedia file.

In addition to **ffmpeg**, the command-line tool for converting multimedia formats, the package includes a streaming server called **ffserver** and a basic player called **ffplay**. Use **ffprobe** to display media information about a file.

Both Mencoder and FFmpeg are complex and you will often find that using internet searching will offer you the command line sequences that you need to achieve a task and those search results will determine the tool you use.

“Ultimately, no GUI app can be as flexible as a comprehensive command line tool.”

Both tools are incredibly versatile. They are driven by a large number of command line arguments and this complexity can, at first, present a barrier to new

users. We'll introduce some of the more common arguments and refer you to the man pages and web-based documentation where you can learn more.

The natural complexity of these tools has brought about several GUI-based tools that try to simplify their use, with varying degrees of success. Ultimately, no GUI application can be as flexible as a comprehensive command line, and these tools present a good case for taking the time to learn their complexities instead of masking them with a GUI.

Codec container

These tools manipulate multimedia files – containers housing one or more streams of encoded data, usually audio and video and sometimes data such as subtitles. Encoding stores audio or video as a bitstream that can later be accessed by a decoder.

The software that performs encoding and decoding is commonly referred to as a codec, and is defined by standards that can have multiple implementations. These implementations all produce acceptable input for any decoder, but may differ in the way they do it.

One popular codec these days is called H.264, otherwise known as MPEG-4 Part 10 or AVC, the Advanced Video Codec. The usual open-source implementation is called x264. H.264 is very good at producing high-quality video with small file sizes and is therefore the current codec of choice for the internet and mobile devices. Blu-ray discs are also encoded using H.264.

The file itself is a container, and there are various container formats that you will encounter including AVI, MP4, MKV and Ogg. You will usually find H.264 within an MP4, MKV or MOV container.

Both Mencoder and FFmpeg use the same libraries for the codecs (**libavcodec**) and containers (**libavformat**).

Enter the command line

So, how do we use them? Basic usage requires that you supply an input file, some options and an output file, like this example:

```
mencoder input-file -oac mp3lame -ovc lavc -o output-file.avi
```

The **mencoder** command reads the **input-file** and uses the given audio and video codecs to re-encode its contents, writing to the specified **output-file**. An **ffmpeg** command line sample to achieve a similar thing would be

```
ffmpeg -i input-file -acodec mp3lame -vcodec lavc -o output-file.avi
```

The **input** can be any file. The formats contained within the file are detected automatically. **ffmpeg** will display the format information if run without output instructions:

```
ffmpeg -i input-file
```

Alternatively, the supplied **ffprobe** command does the same thing:

```
ffprobe input-file
```

As an example, if you can't play our free-and-open Ogg Video files on your smart television then don't fret – just convert them into a MPEG transport stream that your TV will understand. Either

```
mencoder fosspicks_cable3.ogv -oac lavc -ovc lavc -lavcopts vcodec=mpeg2video:acodec=mp2 -of lavf -lavfopts format=mpegts -o fosspicks_cable3.mpg
```

or

```
ffmpeg -i fosspicks_cable3.ogv -vcodec mpeg2video -acodec mp2 -f mpegts fosspicks_cable3.mpg
```

will give you an MPEG2 Transport Stream container that should be playable on your TV, because it uses the same mpeg2video and mp2 audio codecs that broadcast digital TV uses.

Choose your command

These examples show the style of the command-line options for each tool and highlights the subtle differences between them. The main difference is that Mencoder selects the codec library (**-oac** and **-ovc**) and then passes options in to them (**-lavcopts**). Because **ffmpeg** uses the library directly, its arguments are more clearly presented as **ffmpeg** options (**-acodec**, **-vcodec**, **-f**).

You can query Mencoder and FFmpeg to see their supported codecs and container formats:

```
mencoder -oac help -ovc help -of help
```

```
ffmpeg -formats
```

```
ffmpeg -codecs
```

The command line arguments **-oac** and **-ovc** take parameters that select the audio and video codecs to use, but here we instead use **help** to see what's available. The third argument, **-of**, selects the output container format, and the most useful of these is **lavf**, which enables you to use any container format that is supported by FFmpeg's **libavformat** library. Normally the format is inferred from the output filename, but it can be stated explicitly, as we did in our command line examples above.

FFmpeg can display more detailed information about a codec, for example

```
ffmpeg -h encoder=mpeg2video
```

lists all the command line options applicable to that codec. You will see that there are many options available.

You don't have to use an input file. Using the **x11grab** option enables you to capture a screencast directly from your desktop:

```
ffmpeg -f x11grab -r 25 -s 1024x768 -i $DISPLAY -vcodec huffyuv screencast.avi
```

Here we explicitly set the input format and set the source to our X display. The **huffyuv** codec is lossless and captures the screen exactly as-is. We also set a frame rate and screen size. To make the resulting video useful, we should transcode it into a more useful format such as H.264.

```
ffmpeg -i screencast.avi -vcodec libx264 screencast.mp4
```

This time, the input format is detected automatically from the input file. We just need to specify the video codec and output container.

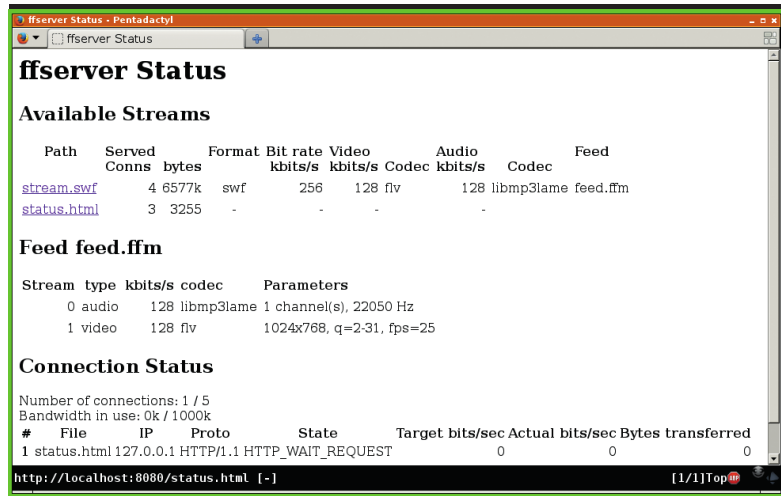
Live video streaming

As well as encoding, FFmpeg can also stream content. Its **ffserver** component accepts feeds from **ffmpeg** clients and streams them over HTTP. Setting up a server requires a simple configuration file called **etc/ffserver.conf**.

```
Port 80
```

```
<Feed feed.ffmpeg>
```

```
File /tmp/feed.ffmpeg
```



ffserver is FFmpeg's streaming server. You can view its status in a browser.

```
</Feed>
<Stream stream.swf>
  Feed feed.ffmpeg
  Format swf
  VideoCodec flv
  VideoBufferSize 80000
  VideoSize 1024x768
  Noaudio
</Stream>
<Stream status.html>
  Format status
</Stream>
```

This minimal example sets up an incoming feed and an outgoing Flash video stream. It also sets up a status page that you can use to see what the server is doing. You run the **ffserver** as a daemon

```
ffserver &
```

Now, you use **ffmpeg** to provide a feed. We'll use our screencast example to provide a live stream of our desktop (you might instead stream from a webcam).

```
ffmpeg -f x11grab -r 25 -s 1024x768 -i $DISPLAY http://localhost/feed.ffmpeg
```

You can then point a web browser at **http://localhost/stream.swf** to see the live streaming video or at **http://localhost/status.html** to view the status of **ffserver**.

With Mencoder and FFmpeg, you can do just about anything you can imagine with multimedia files. They are complicated tools, but they're both well worth making the effort to learn.

“MPlayer and FFmpeg are complicated, but are well worth making the effort to learn.”

John Lane is a technology consultant with a penchant for Linux. He helps new business start-ups make the most of open source.