MPlayer - The Movie Player

http://www.mplayerhq.hu

How to read this documentation

If you are a first-time installer: be sure to read everything from here to the end of the Installation section, and follow the links you will find. If you have any other questions, return to the Table of Contents and search for the topic, read the FAQ, or try grepping through the files. Most questions should be answered somewhere here and the rest has probably already been asked on our mailing lists. Check the archives, there is a lot of valuable information to be found there.

Chapter 1. Introduction

MPlayer is a movie player for Linux (runs on many other Unices, and non-x86 CPUs, see Ports). It plays most MPEG, VOB, AVI, Ogg/OGM, VIVO, ASF/WMA/WMV, QT/MOV/MP4, FLI, RM, NuppelVideo, yuv4mpeg, FILM, RoQ, PVA, Matroska files, supported by many native, XAnim, RealPlayer, and Win32 DLL codecs. You can watch Video CD, SVCD, DVD, 3ivx, RealMedia, Sorenson, Theora, and MPEG-4 (DivX) movies, too. Another big feature of MPlayer is the wide range of supported output drivers. It works with X11, Xv, DGA, OpenGL, SVGAlib, fbdev, AAlib, libcaca, DirectFB, but you can use GGI and SDL (and this way all their drivers) and some low level card-specific drivers (for Matrox, 3Dfx and Radeon, Mach64, Permedia3) too! Most of them support software or hardware scaling, so you can enjoy movies in fullscreen. MPlayer supports displaying through some hardware MPEG decoder boards, such as the DVB and DXR3/Hollywood+. And what about the nice big antialiased shaded subtitles (14 supported types) with European/ISO 8859-1,2 (Hungarian, English, Czech, etc), Cyrillic, Korean fonts, and the onscreen display (OSD)?

The player is rock solid playing damaged MPEG files (useful for some VCDs), and it plays bad AVI files which are unplayable with the famous Windows Media Player. Even AVI files without index chunk are playable, and you can temporarily rebuild their indexes with the `-idx` option, or permanently with MEncoder, thus enabling seeking! As you see, stability and quality are the most important things, but the speed is also amazing. There is also a powerful filter system for video and audio manipulation.

MEncoder (MPlayer's Movie Encoder) is a simple movie encoder, designed to encode MPlayer-playable movies AVI/ASF/OGG/DVD/VCD/VOB/MPG/MOV/VIV/FLI/RM/NUV/NET/PVA to other MPlayer-playable formats (see below). It can encode with various codecs, like MPEG-4 (DivX4) (one or two passes), `libavcodec`, PCM/MP3/VBR MP3 audio.

MEncoder features

- Encoding from the wide range of file formats and decoders of MPlayer
- Encoding to all the codecs of FFmpeg's `libavcodec`
- Video encoding from V4L compatible TV tuners
- Encoding/multiplexing to interleaved AVI files with proper index
- Creating files from external audio stream

- 1, 2 or 3 pass encoding
- VBR MP3 audio
- PCM audio
- Stream copying
- Input A/V synchronizing (pts-based, can be disabled with `-mc 0` option)
- fps correction with `-ofps` option (useful when encoding 30000/1001 fps VOB to 24000/1001 fps AVI)
- Using our very powerful filter system (crop, expand, flip, postprocess, rotate, scale, RGB/YUV conversion)
- Can encode DVD/VOBsub and text subtitles into the output file
- Can rip DVD subtitles to VOBsub format

MPlayer and MEncoder can be distributed under the terms of the GNU General Public License Version 2.

Chapter 2. Installation

A quick installation guide can be found in the `README` file. Please read it first and then come back here for the rest of the gory details.

In this section you will be guided through the compilation and configuration process of MPlayer. It’s not easy, but it won’t necessarily be hard. If you experience a behavior different from this description, please search through this documentation and you’ll find your answers.

2.1. Software requirements

- **POSIX system** - You need a POSIX-compatible shell and POSIX-compatible system tools like grep, sed, awk, etc. in your path.
- **GNU make** 3.81 or later
- **binutils** - GNU binutils 2.11 or later is known to work.
- **compiler** - We mostly use gcc, the recommended versions on x86 are 2.95 and 3.4+. On PowerPC, use 4.x+. icc 10.1+ is also known to work.
- **Xorg/XFree86** - recommended version is 4.3 or later. Make sure the **development packages** are installed, too, otherwise it won’t work. You don’t absolutely need X, some video output drivers work without it.
- **FreeType** - 2.0.9 or later is required for the OSD and subtitles

- **ALSA** - optional, for ALSA audio output support. At least 0.9.0rc4 is required.
- **libjpeg** - required for the optional JPEG video output driver
- **libpng** - required for the optional PNG video output driver
- **directfb** - optional, 0.9.13 or later required for the directfb video output driver
- **lame** - 3.90 or later is recommended, necessary for encoding MP3 audio with MEncoder.
- **zlib** - recommended, many codecs use it.
- **LIVE555 Streaming Media** - optional, needed for some RTSP/RTP streams
- **cdparanoia** - optional, for CDDA support
- **libxmms** - optional, for XMMS input plugin support. At least 1.2.7 is required.
- **libsmb** - optional, for SMB networking support
- **libmad** - optional, for fast integer-only MP3 decoding on FPU-less platforms

2.2. Features

- Decide if you need GUI. If you do, see the GUI section before compiling.
- If you want to install MEncoder (our great all-purpose encoder), see the MEncoder section.
- If you have a V4L compatible **TV tuner** card, and wish to watch/grab and encode movies with MPlayer, read the TV input section.
- If you have a V4L compatible **radio tuner** card, and wish to listen and capture sound with MPlayer, read the radio section.
- There is a neat **OSD Menu** support ready to be used. Check the OSD menu section.

Then build MPlayer:

```
./configure
make
make install
```

At this point, MPlayer is ready to use. Check if you have a `codecs.conf` file in your home directory at (`~/.mplayer/codecs.conf`) left from old MPlayer versions. If you find one, remove it.

Debian users can build a .deb package for themselves, it's very simple. Just exec

```
fakeroot debian/rules binary
```

in MPlayer's root directory. See Debian packaging for detailed instructions.

Always browse the output of `./configure`, and the `configure.log` file, they contain information about what will be built, and what will not. You may also want to view `config.h` and `config.mak` files. If you have some libraries installed, but not detected by `./configure`, then check if you also have the proper header files (usually the -dev packages) and their version matches. The `configure.log` file usually tells you what is missing.

Though not mandatory, the fonts should be installed in order to gain OSD, and subtitle functionality. The recommended method is installing a TTF font file and telling MPlayer to use it. See the Subtitles and OSD section for details.

2.3. What about the GUI?

The GUI needs GTK 1.2.x or GTK 2.0 (it isn't fully GTK, but the panels are), so GTK (and the devel stuff, usually called `gtk-dev`) has to be installed. You can build it by specifying `--enable-gui` during `./configure`. Then, to turn on GUI mode, you have to execute the **gmplayer** binary.

As MPlayer doesn't have a skin included, you have to download one if you want to use the GUI. See the download page. It should be extracted to the usual system-wide directory (`$PREFIX/share/mplayer/skins`), or to `$HOME/.mplayer/skins`. MPlayer by default looks in these directories for a directory named `default`, but you can use the `-skin` *`newskin`* option, or the `skin=newskin` config file directive to use the skin in the `*/skins/newskin` directory.

2.4. Fonts and OSD

You need to tell MPlayer which font to use to enjoy OSD and subtitles. Any TrueType font or special bitmap fonts will work. However, TrueType fonts are recommended as they look far better, can be properly scaled to the movie size and cope better with different encodings.

2.4.1. TrueType fonts

There are two ways to get TrueType fonts to work. The first is to pass the `-font` option to specify a TrueType font file on the command line. This option will be a good candidate to put in your configuration file (see the manual page for details). The second is to create a symlink called `subfont.ttf` to the font file of your choice. Either

```
ln -s /path/to/sample_font.ttf ~/.mplayer/subfont.ttf
```

for each user individually or a system-wide one:

```
ln -s /path/to/sample_font.ttf $PREFIX/share/mplayer/subfont.ttf
```

If MPlayer was compiled with `fontconfig` support, the above methods won't work, instead `-font` expects a `fontconfig` font name and defaults to the sans-serif font. Example:

```
mplayer -font 'Bitstream Vera Sans' anime.mkv
```

To get a list of fonts known to `fontconfig`, use **fc-list**.

2.4.2. bitmap fonts

If for some reason you wish or need to employ bitmap fonts, download a set from our homepage. You can choose between various ISO fonts and some sets of fonts contributed by users in various encodings.

Uncompress the file you downloaded to `~/.mplayer` or `$PREFIX/share/mplayer`. Then rename or symlink one of the extracted directories to `font`, for example:

```
ln -s ~/.mplayer/arial-24 ~/.mplayer/font
```

```
ln -s $PREFIX/share/mplayer/arial-24 $PREFIX/share/mplayer/font
```

Fonts should have an appropriate `font.desc` file which maps Unicode font positions to the actual code page of the subtitle text. Another solution is to have UTF-8-encoded subtitles and use the `-utf8` option or give the subtitles file the same name as your video file with a `.utf` extension and have it in the same directory as the video file.

2.4.3. OSD menu

MPlayer has a completely user-definable OSD Menu interface.

Note

the Preferences menu is currently UNIMPLEMENTED!

Installation

1. compile MPlayer by passing the `--enable-menu` to `./configure`
2. make sure you have an OSD font installed
3. copy `etc/menu.conf` to your `.mplayer` directory
4. copy `etc/input.conf` to your `.mplayer` directory, or to the system-wide MPlayer config dir (default: `/usr/local/etc/mplayer`)
5. check and edit `input.conf` to enable menu movement keys (it is described there).
6. start MPlayer by the following example:

   ```
   mplayer -menu file.avi
   ```

7. push any menu key you defined

2.5. Codec installation

2.5.1. Xvid

Xvid is a free software MPEG-4 ASP compliant video codec. Note that Xvid is not necessary to decode Xvid-encoded video. `libavcodec` is used by default as it offers better speed.

Installing Xvid

Like most open source software, it is available in two flavors: official releases and the CVS version. The CVS version is usually stable enough to use, as most of the time it features fixes for bugs that exist in releases. Here is what to do to make Xvid CVS work with MEncoder:

1. `cvs -z3 -d:pserver:anonymous@cvs.xvid.org:/xvid login`
2. `cvs -z3 -d:pserver:anonymous@cvs.xvid.org:/xvid co xvidcore`
3. `cd xvidcore/build/generic`
4. `./bootstrap.sh && ./configure`

   You may have to add some options (examine the output of **./configure --help**).

5. `make && make install`
6. Recompile MPlayer.

2.5.2. x264

x264 is a library for creating H.264 video. MPlayer sources are updated whenever an x264 API change occurs, so it is always suggested to use MPlayer from Subversion.

If you have a GIT client installed, the latest x264 sources can be gotten with this command:

```
git clone git://git.videolan.org/x264.git
```

Then build and install in the standard way:

```
./configure && make && make install
```

Now rerun `./configure` for MPlayer to pick up `x264` support.

2.5.3. AAC

An open source AAC decoder called FAAD2 is available from http://www.audiocoding.com/downloads.html. MPlayer includes a copy of it in its source tree. If you want to use the external library instead, install it and pass `--enable-faad-external` to `./configure`.

FAAD2 binaries are not available from audiocoding.com, but you can (apt-)get Debian packages from Christian Marillat, Mandrake/Mandriva RPMs from the P.L.F and Fedora/CentOS/RHEL RPMs from RPMFusion.

If you choose to build from source, you do not need all of FAAD2 to decode AAC files, libfaad is enough. Build it like this:

```
cd faad2/
sh bootstrap
./configure
cd libfaad
make
make install
```

2.5.4. AMR

MPlayer can use the OpenCORE AMR libraries through FFmpeg. Download the libraries for AMR-NB and AMR-WB from the opencore-amr project and install them according to the instructions on that page.

2.5.5. XMMS

MPlayer can use XMMS input plugins to play many file formats. There are plugins for SNES game tunes, SID tunes (from Commodore 64), many Amiga formats, .xm, .it, VQF, Musepack, Bonk, shorten and many others. You can find them at the XMMS input plugin page.

For this feature you need to have XMMS and compile MPlayer with `./configure --enable-xmms`.

2.6. RTC

There are three timing methods in MPlayer.

- **To use the old method**, you don't have to do anything. It uses `usleep()` to tune A/V sync, with +/- 10ms accuracy. However sometimes the sync has to be tuned even finer.

- **The new timer** code uses the RTC (RealTime Clock) for this task, because it has precise 1ms timers. The `-rtc` option enables it, but a properly set up kernel is required. If you are running kernel 2.4.19pre8 or later you can adjust the maximum RTC frequency for normal users through the `/proc` file system. Use one of the following two commands to enable RTC for normal users:

  ```
  echo 1024 > /proc/sys/dev/rtc/max-user-freq
  ```

```
sysctl dev/rtc/max-user-freq=1024
```

You can make this setting permanent by adding the latter to `/etc/sysctl.conf`.

You can see the new timer's efficiency in the status line. The power management functions of some notebook BIOSes with speedstep CPUs interact badly with RTC. Audio and video may get out of sync. Plugging the external power connector in before you power up your notebook seems to help. In some hardware combinations (confirmed during usage of non-DMA DVD drive on an ALi1541 board) usage of the RTC timer causes skippy playback. It's recommended to use the third method in these cases.

- **The third timer code** is turned on with the `-softsleep` option. It has the efficiency of the RTC, but it doesn't use RTC. On the other hand, it requires more CPU.

Chapter 3. Usage

3.1. Command line

MPlayer utilizes a complex playtree. Options passed on the command line can apply to all files/URLs or just to specific ones depending on their position. For example

```
mplayer -vfm ffmpeg movie1.avi movie2.avi
```

will use FFmpeg decoders for both files, but

```
mplayer -vfm ffmpeg movie1.avi movie2.avi -vfm dmo
```

will play the second file with a DMO decoder.

You can group filenames/URLs together using `{` and `}`. It is useful with option `-loop`:

```
mplayer { 1.avi -loop 2 2.avi } -loop 3
```

The above command will play files in this order: 1, 1, 2, 1, 1, 2, 1, 1, 2.

Playing a file:

`mplayer` `[`*`options`*`] [`*`path/`*`]`*`filename`*

Another way to play a file:

`mplayer` `[`*`options`*`]` *`file:///uri-escaped-path`*

Playing more files:

`mplayer` `[`*`default options`*`] [`*`path/`*`]`*`filename1`* `[`*`options for filename1`*`]` *`filename2`* `[`*`options for f`*

Playing VCD:

mplayer [*options*] vcd://*trackno* [-cdrom-device */dev/cdrom*]

Playing DVD:

mplayer [*options*] dvd://*titleno* [-dvd-device */dev/dvd*]

Playing from the WWW:

mplayer [*options*] http://*site.com/file.asf*

(playlists can be used, too)

Playing from RTSP:

mplayer [*options*] rtsp://*server.example.com/streamName*

Examples:

```
mplayer -vo x11 /mnt/Films/Contact/contact2.mpg
mplayer vcd://2 -cdrom-device /dev/hdc
mplayer -afm 3 /mnt/DVDtrailers/alien4.vob
mplayer dvd://1 -dvd-device /dev/hdc
mplayer -abs 65536 -delay -0.4 -nobps ~/movies/test.avi
```

3.2. Subtitles and OSD

MPlayer can display subtitles along with movie files. Currently the following formats are supported:

- VOBsub
- OGM
- CC (closed caption)
- MicroDVD
- SubRip
- SubViewer
- Sami
- VPlayer
- RT
- SSA
- PJS (Phoenix Japanimation Society)
- MPsub
- AQTitle
- JACOsub

MPlayer can dump the previously listed subtitle formats (**except the three first**) into the following destination formats, with the given options:

- MPsub: `-dumpmpsub`
- SubRip: `-dumpsrtsub`
- MicroDVD: `-dumpmicrodvdsub`
- JACOsub: `-dumpjacosub`
- Sami: `-dumpsami`

MEncoder can dump DVD subtitles into VOBsub format.

The command line options differ slightly for the different formats:

VOBsub subtitles. VOBsub subtitles consist of a big (some megabytes) `.SUB` file, and optional `.IDX` and/or `.IFO` files. If you have files like *`sample.sub`*, *`sample.ifo`* (optional), *`sample.idx`* - you have to pass MPlayer the `-vobsub sample [-vobsubid id]` options (full path optional). The `-vobsubid` option is like `-sid` for DVDs, you can choose between subtitle tracks (languages) with it. In case that `-vobsubid` is omitted, MPlayer will try to use the languages given by the `-slang` option and fall back to the `langidx` in the `.IDX` file to set the subtitle language. If it fails, there will be no subtitles.

Other subtitles. The other formats consist of a single text file containing timing, placement and text information. Usage: If you have a file like *`sample.txt`*, you have to pass the option `-sub sample.txt` (full path optional).

Adjusting subtitle timing and placement:

`-subdelay sec`

Delays subtitles by *`sec`* seconds. Can be negative. The value is added to movie's time position counter.

`-subfps RATE`

Specify frame/sec rate of subtitle file (float number).

`-subpos 0-100`

Specify the position of subtitles.

If you experience a growing delay between the movie and the subtitles when using a MicroDVD subtitle file, most likely the framerate of the movie and the subtitle file are different. Please note that the MicroDVD subtitle format uses absolute frame numbers for its timing, but there is no fps information in it, and therefore the `-subfps` option should be used with this format. If you like to solve this problem permanently, you have to manually convert the subtitle file framerate. MPlayer can do this conversion for you:

```
mplayer -dumpmicrodvdsub -fps subtitles_fps -subfps avi_fps \
    -sub subtitle_filename dummy.avi
```

About DVD subtitles, read the DVD section.

3.3. Control

MPlayer has a fully configurable, command driven, control layer which lets you control MPlayer with keyboard, mouse, joystick or remote control (using LIRC). See the man page for the complete list of keyboard controls.

3.3.1. Controls configuration

MPlayer allows you bind any key/button to any MPlayer command using a simple config file. The syntax consist of a key name followed by a command. The default config file location is `$HOME/.mplayer/input.conf` but it can be overridden using the `-input conf` option (relative path are relative to `$HOME/.mplayer`).

You can get a full list of supported key names by running **mplayer -input keylist** and a full list of available commands by running **mplayer -input cmdlist**.

Example 3.1. A simple input control file

```
##
## MPlayer input control file
##

RIGHT seek +10
LEFT seek -10
- audio_delay 0.100
+ audio_delay -0.100
q quit
> pt_step 1
< pt_step -1
ENTER pt_step 1 1
```

3.3.2. Control from LIRC

Linux Infrared Remote Control - use an easy to build home-brewed IR-receiver, an (almost) arbitrary remote control and control your Linux box with it! More about it on the LIRC homepage.

If you have the LIRC package installed, `configure` will autodetect it. If everything went fine, MPlayer will print "`Setting up LIRC support...`" on startup. If an error occurs it will tell you. If there is no message about LIRC there is no support compiled in. That's it :-)

The application name for MPlayer is - surprise - `mplayer`. You can use any MPlayer commands and even pass more than one command by separating them with `\n`. Do not forget to enable the repeat flag in `.lircrc` when it makes sense (seek, volume, etc). Here is an excerpt from a sample `.lircrc`:

```
begin
     button = VOLUME_PLUS
     prog = mplayer
     config = volume 1
     repeat = 1
end

begin
    button = VOLUME_MINUS
    prog = mplayer
    config = volume -1
    repeat = 1
end

begin
    button = CD_PLAY
    prog = mplayer
    config = pause
end

begin
    button = CD_STOP
    prog = mplayer
    config = seek 0 1\npause
end
```

If you do not like the standard location for the lirc-config file (`~/.lircrc`) use the `-lircconf` *`filename`* switch to specify another file.

3.3.3. Slave mode

The slave mode allows you to build simple frontends to MPlayer. When run with the `-slave` option MPlayer will read commands separated by a newline (\n) from stdin. The commands are documented in the slave.txt file.

3.4. Streaming from network or pipes

MPlayer can play files from the network, using the HTTP, FTP, MMS or RTSP/RTP protocol.

Playing works simply by passing the URL on the command line. MPlayer honors the `http_proxy` environment variable, using a proxy if available. Proxies can also be forced:

```
mplayer http_proxy://proxy.micorsops.com:3128/http://micorsops.com:80/stream.asf
```

MPlayer can read from stdin (*not* named pipes). This can for example be used to play from FTP:

```
wget ftp://micorsops.com/something.avi -O - | mplayer -
```

Note

It is also recommended to enable `-cache` when playing from the network:

```
wget ftp://micorsops.com/something.avi -O - | mplayer -cache 8192 -
```

3.4.1. Saving streamed content

Once you succeed in making MPlayer play your favorite internet stream, you can use the option `-dumpstream` to save the stream into a file. For example:

```
mplayer http://217.71.208.37:8006 -dumpstream -dumpfile stream.asf
```

will save the content streamed from *`http://217.71.208.37:8006`* into *`stream.asf`*. This works with all protocols supported by MPlayer, like MMS, RTSP, and so forth.

3.5. Edit Decision Lists (EDL)

The edit decision list (EDL) system allows you to automatically skip or mute sections of videos during playback, based on a movie specific EDL configuration file.

This is useful for those who may want to watch a film in "family-friendly" mode. You can cut out any violence, profanity, Jar-Jar Binks .. from a movie according to your own personal preferences. Aside from this, there are other uses, like automatically skipping over commercials in video files you watch.

The EDL file format is pretty bare-bones. There is one command per line that indicates what to do (skip/mute) and when to do it (using pts in seconds).

3.5.1. Using an EDL file

Include the `-edl <filename>` flag when you run MPlayer, with the name of the EDL file you want applied to the video.

3.5.2. Making an EDL file

The current EDL file format is:

```
[begin second] [end second] [action]
```

Where the seconds are floating-point numbers and the action is either `0` for skip or `1` for mute. Example:

```
5.3   7.1    0
15    16.7   1
420   422    0
```

This will skip from second 5.3 to second 7.1 of the video, then mute at 15 seconds, unmute at 16.7 seconds and skip from second 420 to second 422 of the video. These actions will be performed when the playback timer reaches the times given in the file.

To create an EDL file to work from, use the `-edlout <filename>` flag. During playback, just hit i to mark the beginning and end of a skip block. A corresponding entry will be written to the file for that time. You can then go back and fine-tune the generated EDL file as well as change the default operation which is to skip the block described by each line.

Chapter 4. Advanced audio usage

4.1. Surround/Multichannel playback

4.1.1. DVDs

Most DVDs and many other files include surround sound. MPlayer supports surround playback but does not enable it by default because stereo equipment is by far more common. To play a file that has more than two channels of audio use `-channels`. For example, to play a DVD with 5.1 audio:

```
mplayer dvd://1 -channels 6
```

Note that despite the name "5.1" there are actually six discrete channels. If you have surround sound equipment it is safe to put the `channels` option in your MPlayer configuration file `~/.mplayer/config`. For example, to make quadraphonic playback the default, add this line:

```
channels=4
```

MPlayer will then output audio in four channels when all four channels are available.

4.1.2. Playing stereo files to four speakers

MPlayer does not duplicate any channels by default, and neither do most audio drivers. If you want to do that manually:

```
mplayer filename -af channels=2:2:0:1:0:0
```

See the section on channel copying for an explanation.

4.1.3. AC-3/DTS Passthrough

DVDs usually have surround audio encoded in AC-3 (Dolby Digital) or DTS (Digital Theater System) format. Some modern audio equipment is capable of decoding these formats internally. MPlayer can be configured to relay the audio data without decoding it. This will only work if you have a S/PDIF (Sony/Philips Digital Interface) jack in your sound card, or if you are passing audio over HDMI.

If your audio equipment can decode both AC-3 and DTS, you can safely enable passthrough for both formats. Otherwise, enable passthrough for only the format your equipment supports.

To enable passthrough on the command line:

- For AC-3 only, use `-ac hwac3`
- For DTS only, use `-ac hwdts`
- For both AC-3 and DTS, use `-afm hwac3`

To enable passthrough in the MPlayer configuration file:

- For AC-3 only, use `ac=hwac3,`
- For DTS only, use `ac=hwdts,`
- For both AC-3 and DTS, use `afm=hwac3`

Note that there is a comma (",") at the end of `ac=hwac3,` and `ac=hwdts,`. This will make MPlayer fall back on the codecs it normally uses when playing a file that does not have AC-3 or DTS audio. `afm=hwac3` does not need a comma; MPlayer will fall back anyway when an audio family is specified.

4.1.4. MPEG audio Passthrough

Digital TV transmissions (such as DVB and ATSC) and some DVDs usually have MPEG audio streams (in particular MP2). Some MPEG hardware decoders such as full-featured DVB cards and DXR2 adapters can natively decode this format. MPlayer can be configured to relay the audio data without decoding it.

To use this codec:

```
 mplayer -ac hwmpa
```

4.1.5. Matrix-encoded audio

****TODO****

This section has yet to be written and cannot be completed until somebody provides sample files for us to test. If you have any matrix-encoded audio files, know where to find some, or have any information that could be helpful, please send a message to the MPlayer-DOCS mailing list. Put "[matrix-encoded audio]" in the subject line.

If no files or further information are forthcoming this section will be dropped.

Good links:

- http://electronics.howstuffworks.com/surround-sound5.htm
- http://www.extremetech.com/article2/0,1697,1016875,00.asp

4.1.6. Surround emulation in headphones

MPlayer includes an HRTF (Head Related Transfer Function) filter based on an MIT project wherein measurements were taken from microphones mounted on a dummy human head.

Although it is not possible to exactly imitate a surround system, MPlayer's HRTF filter does provide more spatially immersive audio in 2-channel headphones. Regular downmixing simply combines all the channels into two; besides combining the channels, `hrtf` generates subtle echoes, increases the stereo separation slightly, and alters the volume of some frequencies. Whether HRTF sounds better may be dependent on the source audio and a matter of personal taste, but it is definitely worth trying out.

To play a DVD with HRTF:

```
mplayer dvd://1 -channels 6 -af hrtf
```

`hrtf` only works well with 5 or 6 channels. Also, `hrtf` requires 48 kHz audio. DVD audio is already 48 kHz, but if you have a file with a different sampling rate that you want to play using `hrtf` you must resample it:

```
mplayer filename -channels 6 -af resample=48000,hrtf
```

4.1.7. Troubleshooting

If you do not hear any sound out of your surround channels, check your mixer settings with a mixer program such as alsamixer; audio outputs are often muted and set to zero volume by default.

4.2. Channel manipulation

4.2.1. General information

Unfortunately, there is no standard for how channels are ordered. The orders listed below are those of AC-3 and are fairly typical; try them and see if your source matches. Channels are numbered starting with 0.

mono

1. center

stereo

1. left
2. right

quadraphonic

1. left front
2. right front
3. left rear
4. right rear

surround 4.0

1. left front
2. right front
3. center rear
4. center front

surround 5.0

1. left front
2. right front
3. left rear
4. right rear
5. center front

surround 5.1

1. left front
2. right front
3. left rear
4. right rear
5. center front
6. subwoofer

The `-channels` option is used to request the number of channels from the audio decoder. Some audio codecs use the number of specified channels to decide if downmixing the source is necessary. Note that this does not always affect the number of output channels. For example, using `-channels 4` to play a stereo MP3 file will still result in 2-channel output since the MP3 codec will not produce the extra channels.

The `channels` audio filter can be used to create or remove channels and is useful for controlling the number of channels sent to the sound card. See the following sections for more information on channel manipulation.

4.2.2. Playing mono with two speakers

Mono sounds a lot better when played through two speakers - especially when using headphones. Audio files that truly have one channel are automatically played through two speakers; unfortunately, most files with mono sound are actually encoded as stereo with one channel silent. The easiest and most foolproof way to make both speakers output the same audio is the `extrastereo` filter:

```
mplayer filename -af extrastereo=0
```

This averages both channels, resulting in both channels being half as loud as the original. The next sections have examples of other ways to do this without a volume decrease, but they are more complex and require different options depending on which channel to keep. If you really need to maintain the volume, it may be easier to experiment with the `volume` filter and find the right value. For example:

```
mplayer filename -af extrastereo=0,volume=5
```

4.2.3. Channel copying/moving

The `channels` filter can move any or all channels. Setting up all the suboptions for the `channels` filter can be complicated and takes a little care.

1. Decide how many output channels you need. This is the first suboption.
2. Count how many channel moves you will do. This is the second suboption. Each channel can be moved to several different channels at the same time, but keep in mind that when a channel is moved (even if to only one destination) the source channel will be empty unless another channel is moved into it. To copy a channel, keeping the source the

same, simply move the channel into both the destination and the source. For example:

```
channel 2 --> channel 3
channel 2 --> channel 2
```

3. Write out the channel copies as pairs of suboptions. Note that the first channel is 0, the second is 1, etc. The order of these suboptions does not matter as long as they are properly grouped into *`source:destination`* pairs.

Example: one channel in two speakers

Here is an example of another way to play one channel in both speakers. Suppose for this example that the left channel should be played and the right channel discarded. Following the steps above:

1. In order to provide an output channel for each of the two speakers, the first suboption must be "2".
2. The left channel needs to be moved to the right channel, and also must be moved to itself so it won't be empty. This is a total of two moves, making the second suboption "2" as well.
3. To move the left channel (channel 0) into the right channel (channel 1), the suboption pair is "0:1", "0:0" moves the left channel onto itself.

Putting that all together gives:

```
mplayer filename -af channels=2:2:0:1:0:0
```

The advantage this example has over `extrastereo` is that the volume of each output channel is the same as the input channel. The disadvantage is that the suboptions must be changed to "2:2:1:0:1:1" when the desired audio is in the right channel. Also, it is more difficult to remember and type.

Example: left channel in two speakers shortcut

There is actually a much easier way to use the `channels` filter for playing the left channel in both speakers:

```
mplayer filename -af channels=1
```

The second channel is discarded and, with no further suboptions, the single remaining channel is left alone. Sound card drivers automatically play single-channel audio in both speakers. This only works when the desired channel is on the left.

Example: duplicate front channels to the rear

Another common operation is to duplicate the front channels and play them back on the rear speakers of a quadraphonic setup.

1. There should be four output channels. The first suboption is "4".
2. Each of the two front channels needs to be moved to the corresponding rear channel and also to itself. This is four moves, so the second suboption is "4".
3. The left front (channel 0) needs to moved to the left rear (channel 2): "0:2". The left front also needs to be moved to itself: "0:0". The right front (channel 1) is moved to the right rear (channel 3): "1:3", and also to itself: "1:1".

Combine all the suboptions to get:

```
mplayer filename -af channels=4:4:0:2:0:0:1:3:1:1
```

4.2.4. Channel mixing

The `pan` filter can mix channels in user-specified proportions. This allows for everything the `channels` filter can do and more. Unfortunately, the suboptions are much more complicated.

1. Decide how many channels to work with. You may need to specify this with `-channels` and/or `-af channels`. Later examples will show when to use which.

2. Decide how many channels to feed into `pan` (further decoded channels are discarded). This is the first suboption, and it also controls how many channels to employ for output.

3. The remaining suboptions specify how much of each channel gets mixed into each other channel. This is the complicated part. To break the task down, split the suboptions into several sets, one set for each input channel. Each suboption within a set corresponds to an output channel. The number you specify will be the percentage of the input channel that gets mixed into the output channel.

   `pan` accepts values from 0 to 512, yielding 0% to 51200% of the original volume. Be careful when using values greater than 1. Not only can this give you very high volume, but if you exceed the sample range of your sound card you may hear painful pops and clicks. If you want you can follow `pan` with `,volume` to enable clipping, but it is best to keep the values of `pan` low enough that clipping is not necessary.

Example: one channel in two speakers

Here is yet another example for playing the left channel in two speakers. Follow the steps above:

1. `pan` should output two channels, so the first suboption is "2".

2. Since we have two input channels, there will be two sets of suboptions. Since there are also two output channels, there will be two suboptions per set. The left channel from the file should go with full volume to the new left and the right channels. Thus the first set of suboptions is "1:1". The right channel should be discarded, so the second would be "0:0". Any 0 values at the end can be left out, but for ease of understanding we will keep them.

Putting those options together gives:

```
mplayer filename -af pan=2:1:1:0:0
```

If the right channel is desired instead of the left, the suboptions to `pan` will be "2:0:0:1:1".

Example: left channel in two speakers shortcut

As with `channels`, there is a shortcut that only works with the left channel:

```
mplayer filename -af pan=1:1
```

Since `pan` has only one channel of input (the other channel is discarded), there is only one set with one suboption, which specifies that the only channel gets 100% of itself.

Example: downmixing 6-channel PCM

MPlayer's decoder for 6-channel PCM is not capable of downmixing. Here is a way to downmix PCM using `pan`:

1. The number of output channels is 2, so the first suboption is "2".

2. With six input channels there will be six sets of options. Fortunately, since we only care about the output of the first two channels, we only need to make two sets; the remaining four sets can be omitted. Beware that not all multichannel audio files have the same channel order! This example demonstrates downmixing a file with the same channels as AC-3 5.1:

```
0 - front left
1 - front right
2 - rear left
3 - rear right
4 - center front
```

```
5 - subwoofer
```

The first set of suboptions lists the percentages of the original volume, in order, which each output channel should receive from the front left channel: "1:0". The front right channel should go into the right output: "0:1". The same for the rear channels: "1:0" and "0:1". The center channel goes into both output channels with half volume: "0.5:0.5", and the subwoofer goes into both with full volume: "1:1".

Put all that together, for:

```
mplayer 6-channel.wav -af pan=2:1:0:0:1:1:0:0:1:0.5:0.5:1:1
```

The percentages listed above are only a rough example. Feel free to tweak them.

Example: Playing 5.1 audio on big speakers without a subwoofer

If you have a huge pair of front speakers you may not want to waste any money on buying a subwoofer for a complete 5.1 sound system. If you use `-channels 5` to request that liba52 decode 5.1 audio in 5.0, the subwoofer channel is simply discarded. If you want to distribute the subwoofer channel yourself you need to downmix manually with `pan`:

1. Since `pan` needs to examine all six channels, specify `-channels 6` so liba52 decodes them all.
2. `pan` outputs to only five channels, the first suboption is 5.
3. Six input channels and five output channels means six sets of five suboptions.
   - The left front channel only replicates onto itself: "1:0:0:0:0"
   - Same for the right front channel: "0:1:0:0:0"
   - Same for the left rear channel: "0:0:1:0:0"
   - And also the same for the right rear channel: "0:0:0:1:0"
   - Center front, too: "0:0:0:0:1"
   - And now we have to decide what to do with the subwoofer, e.g. half into front right and front left: "0.5:0.5:0:0:0"

Combine all those options to get:

```
mplayer dvd://1 -channels 6 -af pan=5:1:0:0:0:0:0:1:0:0:0:0:0:1:0:0:0:0:0:1:0:0:0:0:0:1:0.
```

4.3. Software Volume adjustment

Some audio tracks are too quiet to be heard comfortably without amplification. This becomes a problem when your audio equipment cannot amplify the signal for you. The `-softvol` option directs MPlayer to use an internal mixer. You can then use the volume adjustment keys (by default **9** and **0**) to reach much higher volume levels. Note that this does not bypass your sound card's mixer; MPlayer only amplifies the signal before sending it to your sound card. The following example is a good start:

```
mplayer quiet-file -softvol -softvol-max 300
```

The `-softvol-max` option specifies the maximum allowable output volume as a percentage of the original volume. For example, `-softvol-max 200` would allow the volume to be adjusted up to twice its original level. It is safe to specify a large value with `-softvol-max`; the higher volume will not be used until you use the volume adjustment keys. The only disadvantage of a large value is that, since MPlayer adjusts volume by a percentage of the maximum, you will not have as precise control when using the volume adjustment keys. Use a lower value with `-softvol-max` and/or specify `-volstep 1` if you need higher precision.

The `-softvol` option works by controlling the `volume` audio filter. If you want to play a file at a certain volume from the beginning you can specify `volume` manually:

```
mplayer quiet-file -af volume=10
```

This will play the file with a ten decibel gain. Be careful when using the `volume` filter - you could easily hurt your ears if you use too high a value. Start low and work your way up gradually until you get a feel for how much adjustment is required. Also, if you specify excessively high values, `volume` may need to clip the signal to avoid sending your sound card data that is outside the allowable range; this will result in distorted audio.

Chapter 5. CD/DVD usage

5.1. CD/DVD drives

Modern CD-ROM drives can attain very high head speeds, yet some CD-ROM drives are capable of running at reduced speeds. There are several reasons that might make you consider changing the speed of a CD-ROM drive:

- There have been reports of read errors at high speeds, especially with badly pressed CD-ROMs. Reducing the speed can prevent data loss under these circumstances.
- Many CD-ROM drives are annoyingly loud, a lower speed may reduce the noise.

5.1.1. Linux

You can reduce the speed of IDE CD-ROM drives with **hdparm**, **setcd** or **cdctl**. It works like this:

```
hdparm -E [speed] [cdrom device]
```

```
setcd -x [speed] [cdrom device]
```

```
cdctl -bS [speed]
```

If you are using SCSI emulation, you might have to apply the settings to the real IDE device, not the emulated SCSI device.

If you have root privileges the following command may also help:

```
echo file_readahead:2000000 > /proc/ide/[cdrom device]/settings
```

This sets prefetched file reading to 2MB, which helps with scratched CD-ROMs. If you set it to too high, the drive will continuously spin up and down, and will dramatically decrease the performance. It is recommended that you also tune your CD-ROM drive with **hdparm**:

```
hdparm -d1 -a256 -u1 [cdrom device]
```

This enables DMA access, read-ahead, and IRQ unmasking (read the **hdparm** man page for a detailed explanation).

Please refer to "`/proc/ide/[cdrom device]/settings`" for fine-tuning your CD-ROM.

You may tweak the speed of SCSI CD-ROM drives with **sdparm**, you need version 1.03 or higher:

```
sdparm --command=speed=[speed in kB/s] [cdrom device]
```

Speed must be specified in kilobytes per second, the drive will round it as appropriate. Please refer to the sdparm man page for details.

There is also a dedicated tool that works for Plextor SCSI drives.

5.1.2. FreeBSD

speed:

```
cdcontrol [-f device] speed [speed]
```

DMA:

```
sysctl hw.ata.atapi_dma=1
```

5.2. DVD playback

For the complete list of available options, please read the man page. The syntax to play a standard DVD is as follows:

```
mplayer dvd://<track> [-dvd-device <device>]
```

Example:

```
mplayer dvd://1 -dvd-device /dev/hdc
```

If you have compiled MPlayer with dvdnav support, the syntax is the same, except that you need to use dvdnav:// instead of dvd://.

The default DVD device is `/dev/dvd`. If your setup differs, make a symlink or specify the correct device on the command line with the `-dvd-device` option.

MPlayer uses `libdvdread` and `libdvdcss` for DVD playback and decryption. These two libraries are contained in the MPlayer source tree, you do not have to install them separately. You can also use system-wide versions of the two libraries, but this solution is not recommended, as it can result in bugs, library incompatibilities and slower speed.

Note

In case of DVD decoding problems, try disabling supermount, or any other such facilities. Some RPC-2 drives may also require setting the region code.

DVD decryption. DVD decryption is done by `libdvdcss`. The method can be specified through the `DVDCSS_METHOD` environment variable, see the manual page for details.

5.2.1. region code

DVD drives nowadays come with a nonsensical restriction labeled region code. This is a scheme to force DVD drives to only accept DVDs produced for one of the six different regions into which the world was partitioned. How a group of people can sit around a table, come up with such an idea and expect the world of the 21st century to bow to their will is beyond anyone's guess.

Drives that enforce region settings through software only are also known as RPC-1 drives, those that do it in hardware as RPC-2. RPC-2 drives allow changing the region code five times before it remains fixed. Under Linux you can use the regionset tool to set the region code of your DVD drive.

Thankfully, it is possible to convert RPC-2 drives into RPC-1 drives through a firmware upgrade. Feed the model number of your DVD drive into your favorite search engine or have a look at the forum and download sections of "The firmware page". While the usual caveats for firmware upgrades apply, experience with getting rid of region code enforcement is generally positive.

5.3. VCD playback

For the complete list of available options, please read the man page. The Syntax for a standard Video CD (VCD) is as follows:

```
mplayer vcd://<track> [-cdrom-device <device>]
```

Example:

```
mplayer vcd://2 -cdrom-device /dev/hdc
```

The default VCD device is `/dev/cdrom`. If your setup differs, make a symlink or specify the correct device on the command line with the `-cdrom-device` option.

Note

At least Plextor and some Toshiba SCSI CD-ROM drives have horrible performance reading VCDs. This is because the CDROMREADRAW `ioctl` is not fully implemented for these drives. If you have some knowledge of SCSI programming, please help us implement generic SCSI support for VCDs.

In the meantime you can extract data from VCDs with readvcd and play the resulting file with MPlayer.

VCD structure. A Video CD (VCD) is made up of CD-ROM XA sectors, i.e. CD-ROM mode 2 form 1 and 2 tracks:

- The first track is in mode 2 form 2 format which means it uses L2 error correction. The track contains an ISO-9660 file system with 2048 bytes/sector. This file system contains VCD metadata information, as well as still frames often used in menus. MPEG segments for menus can also be stored in this first track, but the MPEGs have to be broken up into a series of 150-sector chunks. The ISO-9660 file system may contain other files or programs that are not essential for VCD operation.
- The second and remaining tracks are generally raw 2324 bytes/sector MPEG (movie) tracks, containing one MPEG PS data packet per sector. These are in mode 2 form 1 format, so they store more data per sector at the loss of some error correction. It is also legal to have CD-DA tracks in a VCD after the first track as well. On some operating systems there is some trickery that goes on to make these non-ISO-9660 tracks appear in a file system. On other operating systems like GNU/Linux this is not the case (yet). Here the MPEG data **cannot be mounted**. As most movies are inside this kind of track, you should try `vcd://2` first.
- There exist VCD disks without the first track (single track and no file system at all). They are still playable, but cannot be mounted.
- The definition of the Video CD standard is called the Philips "White Book" and it is not generally available online as it must be purchased from Philips. More detailed information about Video CDs can be found in the vcdimager documentation.

About .DAT files. The ~600 MB file visible on the first track of the mounted VCD is not a real file! It is a so called ISO gateway, created to allow Windows to handle such tracks (Windows does not allow raw device access to applications at all). Under Linux you cannot copy or play such files (they contain garbage). Under Windows it is possible as its iso9660 driver emulates the raw reading of tracks in this file. To play a .DAT file you need the kernel driver which can be found in the Linux version of PowerDVD. It has a modified iso9660 file system (`vcdfs/isofs-2.4.X.o`) driver, which is able to emulate the raw tracks through this shadow .DAT file. If you mount the disc using their driver, you can copy and even play .DAT files with MPlayer. But it will not work with the standard iso9660 driver of the Linux kernel! Use `vcd://` instead. Alternatives for VCD copying are the new cdfs kernel driver (not part of the official kernel) that shows CD sessions as image files and cdrdao, a bit-by-bit CD grabbing/copying application.

Chapter 6. TV

6.1. TV input

This section is about how to enable **watching/grabbing from V4L compatible TV tuner**. See the man page for a description of TV options and keyboard controls.

6.1.1. Compilation

1. First, you have to recompile. `./configure` will autodetect kernel headers of v4l stuff and the existence of `/dev/video*` entries. If they exist, TV support will be built (see the output of `./configure`).
2. Make sure your tuner works with another TV software in Linux, for example XawTV.

6.1.2. Usage tips

The full listing of the options is available on the manual page. Here are just a few tips:

- Use the `channels` option. An example:

  ```
  -tv channels=26-MTV1,23-TV2
  ```

  Explanation: Using this option, only the 26 and 23 channels will be usable, and there will be a nice OSD text upon channel switching, displaying the channel’s name. Spaces in the channel name must be replaced by the "_" character.

- Choose some sane image dimensions. The dimensions of the resulting image should be divisible by 16.

- If you capture the video with the vertical resolution higher than half of the full resolution (i.e. 288 for PAL or 240 for NTSC), then the ’frames’ you get will really be interleaved pairs of fields. Depending on what you want to do with the video you may leave it in this form, destructively deinterlace, or break the pairs apart into individual fields.

  Otherwise you’ll get a movie which is distorted during fast-motion scenes and the bitrate controller will be probably even unable to retain the specified bitrate as the interlacing artifacts produce high amount of detail and thus consume lot of bandwidth. You can enable deinterlacing with `-vf pp=DEINT_TYPE`. Usually `pp=lb` does a good job, but it can be matter of personal preference. See other deinterlacing algorithms in the manual and give it a try.

- Crop out the dead space. When you capture the video, the areas at the edges are usually black or contain some noise. These again consume lots of unnecessary bandwidth. More precisely it’s not the black areas themselves but the sharp transitions between the black and the brighter video image which do but that’s not important for now. Before you start capturing, adjust the arguments of the `crop` option so that all the crap at the margins is cropped out. Again, don’t forget to keep the resulting dimensions sane.

- Watch out for CPU load. It shouldn’t cross the 90% boundary for most of the time. If you have a large capture buffer, MEncoder can survive an overload for few seconds but nothing more. It’s better to turn off the 3D OpenGL screensavers and similar stuff.

- Don’t mess with the system clock. MEncoder uses the system clock for doing A/V sync. If you adjust the system clock (especially backwards in time), MEncoder gets confused and you will lose frames. This is an important issue if you are hooked to a network and run some time synchronization software like NTP. You have to turn NTP off during the capture process if you want to capture reliably.

- Don’t change the `outfmt` unless you know what you are doing or your card/driver really doesn’t support the default (YV12 colorspace). In the older versions of MPlayer/ MEncoder it was necessary to specify the output format. This issue should be fixed in the current releases and `outfmt` isn't required anymore, and the default suits the most purposes. For example, if you are capturing into DivX using `libavcodec` and specify `outfmt=RGB24` in order to increase the quality of the captured images, the captured image will be actually later converted back into YV12 so the only thing you achieve is a massive waste of CPU power.

- There are several ways of capturing audio. You can grab the sound either using your sound card via an external cable connection between video card and line-in, or using the built-in ADC in the bt878 chip. In the latter case, you have to load the **btaudio** driver. Read the `linux/Documentation/sound/btaudio` file (in the kernel tree, not MPlayer's) for some instructions on using this driver.

- If MEncoder cannot open the audio device, make sure that it is really available. There can be some trouble with the sound servers like aRts (KDE) or ESD (GNOME). If you have a full duplex sound card (almost any decent card supports it today), and you are using KDE, try to check the "full duplex" option in the sound server preference menu.

6.1.3. Examples

Dummy output, to AAlib :)

```
mplayer -tv driver=dummy:width=640:height=480 -vo aa tv://
```

Input from standard V4L:

```
mplayer -tv driver=v4l:width=640:height=480:outfmt=i420 -vc rawi420 -vo xv tv://
```

A more sophisticated example. This makes MEncoder capture the full PAL image, crop the margins, and deinterlace the picture using a linear blend algorithm. Audio is compressed with a constant bitrate of 64kbps, using LAME codec. This setup is suitable for capturing movies.

```
mencoder -tv driver=v4l:width=768:height=576 -oac mp3lame -lameopts cbr:br=64\
    -ovc lavc -lavcopts vcodec=mpeg4:vbitrate=900 \
    -vf crop=720:544:24:16,pp=lb -o output.avi tv://
```

This will additionally rescale the image to 384x288 and compresses the video with the bitrate of 350kbps in high quality mode. The vqmax option looses the quantizer and allows the video compressor to actually reach so low bitrate even at the expense of the quality. This can be used for capturing long TV series, where the video quality isn't so important.

```
mencoder -tv driver=v4l:width=768:height=576 \
    -ovc lavc -lavcopts vcodec=mpeg4:vbitrate=350:vhq:vqmax=31:keyint=300 \
    -oac mp3lame -lameopts cbr:br=48 -sws 1 -o output.avi\
    -vf crop=720:540:24:18,pp=lb,scale=384:288 tv://
```

It's also possible to specify smaller image dimensions in the `-tv` option and omit the software scaling but this approach uses the maximum available information and is a little more resistant to noise. The bt8x8 chips can do the pixel averaging only in the horizontal direction due to a hardware limitation.

6.2. Teletext

Teletext is currently available only in MPlayer for v4l and v4l2 drivers.

6.2.1. Implementation notes

MPlayer supports regular text, graphics and navigation links. Unfortunately, colored pages are not fully supported yet - all pages are shown as grayscaled. Subtitle pages (also known as Closed Captions) are supported, too.

MPlayer starts caching all teletext pages upon starting to receive TV input, so you do not need to wait until the requested page is loaded.

Note: Using teletext with `-vo xv` causes strange colors.

6.2.2. Using teletext

To enable teletext decoding you must specify the VBI device to get teletext data from (usually `/dev/vbi0` for Linux). This can be done by specifying `tdevice` in your configuration file, like shown below:

```
tv=tdevice=/dev/vbi0
```

You might need to specify the teletext language code for your country. To list all available country codes use

```
tv=tdevice=/dev/vbi0:tlang=-1
```

Here is an example for Russian:

```
tv=tdevice=/dev/vbi0:tlang=33
```

6.2.3. Teletext hot keys

Key	Description
X	Switch teletext display on/off
C	Cycle through teletext rendering modes (opaque, transparent, inverted opaque, inverted transparent
Left/Right	Go to previous/next teletext page
PageUp/PageDown	Go to next/previous teletext subpage
digits	Enter teletext page number to jump to

Chapter 7. Radio

7.1. Radio input

This section is about how to enable listening to radio from a V4L-compatible radio tuner. See the man page for a description of radio options and keyboard controls.

7.1.1. Compilation

1. First, you have to recompile MPlayer using `./configure` with `--enable-radio` and (if you want capture support) `--enable-radio-capture`.

2. Make sure your tuner works with another radio software in Linux, for example XawTV.

7.1.2. Usage tips

The full listing of the options is available in the manual page. Here are just a few tips:

- Use the `channels` option. An example:

  ```
  -radio channels=104.4-Sibir,103.9-Maximum
  ```

  Explanation: With this option, only the 104.4 and 103.9 radio stations will be usable. There will be a nice OSD text upon channel switching, displaying the channel's name. Spaces in the channel name must be replaced by the "_" character.

- There are several ways of capturing audio. You can grab the sound either using your sound card via an external cable connection between video card and line-in, or using the built-in ADC in the saa7134 chip. In the latter case, you have to load the `saa7134-alsa` or `saa7134-oss` driver.

- MEncoder cannot be used for audio capture, because it requires a video stream to work. So your can either use arecord from ALSA project or use `-ao pcm:file=file.wav`. In the latter case you will not hear any sound (unless you are using a line-in cable and have switched line-in mute off).

7.1.3. Examples

Input from standard V4L (using line-in cable, capture switched off):

```
mplayer radio://104.4
```

Input from standard V4L (using line-in cable, capture switched off, V4Lv1 interface):

```
mplayer -radio driver=v4l radio://104.4
```

Playing second channel from channel list:

```
mplayer -radio channels=104.4=Sibir,103.9=Maximm radio://2
```

Passing sound over the PCI bus from the radio card's internal ADC. In this example the tuner is used as a second sound card (ALSA device hw:1,0). For saa7134-based cards either the `saa7134-alsa` or `saa7134-oss` module must be loaded.

```
mplayer -rawaudio rate=32000 radio://2/capture \
    -radio adevice=hw=1.0:arate=32000:channels=104.4=Sibir,103.9=Maximm
```

Note

When using ALSA device names colons must be replaced by equal signs, commas by periods.

Chapter 8. Video output devices

8.1. Setting up MTRR

It is VERY recommended to check if the MTRR registers are set up properly, because they can give a big performance boost.

Do a **cat /proc/mtrr**:

```
--($:~)-- cat /proc/mtrr
reg00: base=0xe4000000 (3648MB), size=  16MB: write-combining, count=9
reg01: base=0xd8000000 (3456MB), size= 128MB: write-combining, count=1
```

It's right, shows my Matrox G400 with 16MB memory. I did this from XFree 4.x.x, which sets up MTRR registers automatically.

If nothing worked, you have to do it manually. First, you have to find the base address. You have 3 ways to find it:

1. from X11 startup messages, for example:

   ```
   (--) SVGA: PCI: Matrox MGA G400 AGP rev 4, Memory @ 0xd8000000, 0xd4000000
   (--) SVGA: Linear framebuffer at 0xD8000000
   ```

2. from `/proc/pci` (use **lspci -v** command):

   ```
   01:00.0 VGA compatible controller: Matrox Graphics, Inc.: Unknown device 0525
   Memory at d8000000 (32-bit, prefetchable)
   ```

3. from mga_vid kernel driver messages (use **dmesg**):

   ```
   mga_mem_base = d8000000
   ```

Then let's find the memory size. This is very easy, just convert video RAM size to hexadecimal, or use this table:

1 MB	0x100000
2 MB	0x200000
4 MB	0x400000
8 MB	0x800000

16 MB	0x1000000
32 MB	0x2000000

You know base address and memory size, let's setup MTRR registers! For example, for the Matrox card above (`base=0xd8000000`) with 32MB ram (`size=0x2000000`) just execute:

```
echo "base=0xd8000000 size=0x2000000 type=write-combining" > /proc/mtrr
```

Not all CPUs have MTRRs. For example older K6-2 (around 266MHz, stepping 0) CPUs don't have MTRRs, but stepping 12 does (execute **cat /proc/cpuinfo** to check it).

8.2. Xv

Under XFree86 4.0.2 or newer, you can use your card's hardware YUV routines using the XVideo extension. This is what the option `-vo xv` uses. Also, this driver supports adjusting brightness/contrast/hue/etc. (unless you use the old, slow DirectShow DivX codec, which supports it everywhere), see the man page.

In order to make this work, be sure to check the following:

1. You have to use XFree86 4.0.2 or newer (former versions don't have XVideo)

2. Your card actually supports hardware acceleration (modern cards do)

3. X loads the XVideo extension, it's something like this:

   ```
   (II) Loading extension XVideo
   ```

   in `/var/log/XFree86.0.log`

   ### Note

   This loads only the XFree86's extension. In a good install, this is always loaded, and doesn't mean that the **card's** XVideo support is loaded!

4. Your card has Xv support under Linux. To check, try **xvinfo**, it is the part of the XFree86 distribution. It should display a long text, similar to this:

   ```
   X-Video Extension version 2.2
   screen #0
     Adaptor #0: "Savage Streams Engine"
       number of ports: 1
       port base: 43
       operations supported: PutImage
       supported visuals:
         depth 16, visualID 0x22
         depth 16, visualID 0x23
       number of attributes: 5
   (...)
       Number of image formats: 7
         id: 0x32595559 (YUY2)
           guid: 59555932-0000-0010-8000-00aa00389b71
           bits per pixel: 16
           number of planes: 1
           type: YUV (packed)
         id: 0x32315659 (YV12)
           guid: 59563132-0000-0010-8000-00aa00389b71
           bits per pixel: 12
           number of planes: 3
           type: YUV (planar)
   (...etc...)
   ```

It must support YUY2 packed, and YV12 planar pixel formats to be usable with MPlayer.

5. And finally, check if MPlayer was compiled with 'xv' support. Do a **mplayer -vo help | grep xv** . If 'xv' support was built a line similar to this should appear:

```
    xv       X11/Xv
```

8.2.1. 3dfx cards

Older 3dfx drivers were known to have problems with XVideo acceleration, it didn't support YUY2 or YV12 colorspaces. Verify that you have XFree86 version 4.2.0 or later, it can handle YV12 and YUY2 while previous versions, including 4.1.0, **crash with YV12**. If you experience strange effects using `-vo xv`, try SDL (it has XVideo, too) and see if it helps. Check the SDL section for details.

OR, try the NEW `-vo tdfxfb` driver! See the tdfxfb section.

8.2.2. S3 cards

S3 Savage3D's should work fine, but for Savage4, use XFree86 version 4.0.3 or greater (in case of image problems, try 16bpp). As for S3 Virge: there is xv support, but the card itself is very slow, so you better sell it.

There is now a native framebuffer driver for S3 Virge cards similar to tdfxfb. Set up your framebuffer (e.g. append "`vga=792 video=vesa:mtrr`" to your kernel command line) and use `-vo s3fb` (`-vf yuy2` and `-dr` will also help).

Note

It's currently unclear which Savage models lack YV12 support, and convert by driver (slow). If you suspect your card, get a newer driver, or ask politely on the MPlayer-users mailing list for an MMX/3DNow! enabled driver.

8.2.3. nVidia cards

nVidia isn't always a very good choice under Linux ... XFree86's open-source driver supports most of these cards, but for some cases, you'll have to use the binary closed-source nVidia driver, available at nVidia's web site. You'll always need this driver if you want 3D acceleration, too.

Riva128 cards don't have XVideo support with XFree86's nVidia driver :(Complain to nVidia.

However, MPlayer contains a VIDIX driver for most nVidia cards. Currently it is in beta stage, and has some drawbacks. For more information, see nVidia VIDIX section.

8.2.4. ATI cards

The GATOS driver (which you should use, unless you have Rage128 or Radeon) has VSYNC enabled by default. It means that decoding speed (!) is synced to the monitor's refresh rate. If playing seems to be slow, try disabling VSYNC somehow, or set refresh rate to a n*(fps of the movie) Hz.

Radeon VE - if you need X, use XFree86 4.2.0 or greater for this card. No TV out support. Of course with MPlayer you can happily get **accelerated** display, with or without **TV output**, and no libraries or X are needed. Read the VIDIX section.

8.2.5. NeoMagic cards

These cards can be found in many laptops. You must use XFree86 4.3.0 or above, or else use Stefan Seyfried's Xv-capable drivers. Just choose the one that applies to your version of XFree86.

XFree86 4.3.0 includes Xv support, yet Bohdan Horst sent a small patch against the XFree86 sources that speeds up framebuffer operations (so XVideo) up to four times. The patch has been included in XFree86 CVS and should be in the next release after 4.3.0.

To allow playback of DVD sized content change your XF86Config like this:

```
Section "Device"
    [...]
    Driver "neomagic"
    Option "OverlayMem" "829440"
    [...]
EndSection
```

8.2.6. Trident cards

If you want to use Xv with a Trident card, provided that it doesn't work with 4.1.0, install XFree 4.2.0. 4.2.0 adds support for fullscreen Xv support with the Cyberblade XP card.

Alternatively, MPlayer contains a VIDIX driver for the Cyberblade/i1 card.

8.2.7. Kyro/PowerVR cards

If you want to use Xv with a Kyro based card (for example Hercules Prophet 4000XT), you should download the drivers from the PowerVR site.

8.2.8. Intel cards

These cards can be found in many laptops. Recent Xorg is recommended.

To allow playback of DVD sized (and larger) content change your XF86Config/xorg.conf like this:

```
Section "Device"
    [...]
    Driver "intel"
    Option "LinearAlloc" "6144"
    [...]
EndSection
```

Lack of this option usually results in an error like

```
X11 error: BadAlloc (insufficient resources for operation)
```

when attempting to use `-vo xv`.

8.3. DGA

PREAMBLE. This document tries to explain in some words what DGA is in general and what the DGA video output driver for MPlayer can do (and what it can't).

WHAT IS DGA. DGA is short for *Direct Graphics Access* and is a means for a program to bypass the X server and directly modifying the framebuffer memory. Technically spoken this happens by mapping the framebuffer memory into the memory range of your process. This is allowed by the kernel only if you have superuser privileges. You can get these either by logging in as `root` or by setting the SUID bit on the MPlayer executable (**not recommended**).

There are two versions of DGA: DGA1 is used by XFree 3.x.x and DGA2 was introduced with XFree 4.0.1.

DGA1 provides only direct framebuffer access as described above. For switching the resolution of the video signal you have to rely on the XVidMode extension.

DGA2 incorporates the features of XVidMode extension and also allows switching the depth of the display. So you may, although basically running a 32 bit depth X server, switch to a depth of 15 bits and vice versa.

However DGA has some drawbacks. It seems it is somewhat dependent on the graphics chip you use and on the implementation of the X server's video driver that controls this chip. So it does not work on every system...

INSTALLING DGA SUPPORT FOR MPLAYER. First make sure X loads the DGA extension, see in `/var/log/XFree86.0.log`:

```
(II) Loading extension XFree86-DGA
```

See, XFree86 4.0.x or greater is **highly recommended**! MPlayer's DGA driver is autodetected by `./configure`, or you can force it with `--enable-dga`.

If the driver couldn't switch to a smaller resolution, experiment with options `-vm` (only with X 3.3.x), `-fs`, `-bpp`, `-zoom` to find a video mode that the movie fits in. There is no converter right now :(

Become `root`. DGA needs root access to be able to write directly video memory. If you want to run it as user, then install MPlayer SUID root:

```
chown root /usr/local/bin/mplayer
chmod 750 /usr/local/bin/mplayer
chmod +s /usr/local/bin/mplayer
```

Now it works as a simple user, too.

Security risk

This is a **big** security risk! **Never** do this on a server or on a computer that can be accessed by other people because they can gain root privileges through SUID root MPlayer.

Now use `-vo dga` option, and there you go! (hope so:) You should also try if the `-vo sdl:driver=dga` option works for you! It's much faster!

RESOLUTION SWITCHING. The DGA driver allows for switching the resolution of the output signal. This avoids the need for doing (slow) software scaling and at the same time provides a fullscreen image. Ideally it would switch to the exact resolution (except for honoring aspect ratio) of the video data, but the X server only allows switching to resolutions predefined in `/etc/X11/XF86Config` (`/etc/X11/XF86Config-4` for XFree 4.X.X respectively). Those are defined by so-called modelines and depend on the capabilities of your video hardware. The X server scans this config file on startup and disables the modelines not suitable for your hardware. You can find out which modes survive with the X11 log file. It can be found at: `/var/log/XFree86.0.log`.

These entries are known to work fine with a Riva128 chip, using the nv.o X server driver module.

```
Section "Modes"
  Identifier "Modes[0]"
  Modeline "800x600"  40     800 840 968 1056  600 601 605 628
  Modeline "712x600"  35.0   712 740 850 900   400 410 412 425
  Modeline "640x480"  25.175 640 664 760 800   480 491 493 525
  Modeline "400x300"  20     400 416 480 528   300 301 303 314 Doublescan
  Modeline "352x288"  25.10  352 368 416 432   288 296 290 310
  Modeline "352x240"  15.750 352 368 416 432   240 244 246 262 Doublescan
  Modeline "320x240"  12.588 320 336 384 400   240 245 246 262 Doublescan
EndSection
```

DGA & MPLAYER. DGA is used in two places with MPlayer: The SDL driver can be made to make use of it (`-vo sdl:driver=dga`) and within the DGA driver (`-vo dga`). The above said is true for both; in the following sections I'll explain how the DGA driver for MPlayer works.

FEATURES. The DGA driver is invoked by specifying `-vo dga` at the command line. The default behavior is to switch to a resolution matching the original resolution of the video as close as possible. It deliberately ignores the `-vm` and `-fs` options (enabling of video mode switching and fullscreen) - it always tries to cover as much area of your screen as possible by switching the video mode, thus refraining from using additional cycles of your CPU to scale the image. If you don't like the mode it chooses you may force it to choose the mode matching closest the resolution you specify by `-x` and `-y`. By providing the `-v` option, the DGA driver will print, among a lot of other things, a list of all resolutions supported by your current `XF86Config` file. Having DGA2 you may also force it to use a certain depth by using the `-bpp` option. Valid depths are 15, 16, 24 and 32. It depends on your hardware whether these depths are natively supported or if a (possibly slow) conversion has to be done.

If you should be lucky enough to have enough offscreen memory left to put a whole image there, the DGA driver will use double buffering, which results in much smoother movie playback. It will tell you whether double buffering is enabled or not.

Double buffering means that the next frame of your video is being drawn in some offscreen memory while the current frame is being displayed. When the next frame is ready, the graphics chip is just told the location in memory of the new frame and simply fetches the data to be displayed from there. In the meantime the other buffer in memory will be filled again with new video data.

Double buffering may be switched on by using the option `-double` and may be disabled with `-nodouble`. Current default option is to disable double buffering. When using the DGA driver, onscreen display (OSD) only works with double buffering enabled. However, enabling double buffering may result in a big speed penalty (on my K6-II+ 525 it used an additional 20% of CPU time!) depending on the implementation of DGA for your hardware.

SPEED ISSUES. Generally spoken, DGA framebuffer access should be at least as fast as using the X11 driver with the additional benefit of getting a fullscreen image. The percentage speed values printed by MPlayer have to be interpreted with some care, as for example, with the X11 driver they do not include the time used by the X server needed for the actual drawing. Hook a terminal to a serial line of your box and start **top** to see what is really going on in your box.

Generally spoken, the speedup done by using DGA against 'normal' use of X11 highly depends on your graphics card and how well the X server module for it is optimized.

If you have a slow system, better use 15 or 16 bit depth since they require only half the memory bandwidth of a 32 bit display.

Using a depth of 24 bit is a good idea even if your card natively just supports 32 bit depth since it transfers 25% less data compared to the 32/32 mode.

I've seen some AVI files be played back on a Pentium MMX 266. AMD K6-2 CPUs might work at 400 MHZ and above.

KNOWN BUGS. Well, according to some developers of XFree, DGA is quite a beast. They tell you better not to use it. Its implementation is not always flawless with every chipset driver for XFree out there.

- With XFree 4.0.3 and `nv.o` there is a bug resulting in strange colors.
- ATI driver requires to switch mode back more than once after finishing using of DGA.
- Some drivers simply fail to switch back to normal resolution (use **Ctrl**+**Alt**+**Keypad +** and **Ctrl**+**Alt**+**Keypad -** to switch back manually).
- Some drivers simply display strange colors.
- Some drivers lie about the amount of memory they map into the process's address space, thus vo_dga won't use double buffering (SIS?).
- Some drivers seem to fail to report even a single valid mode. In this case the DGA driver will crash telling you about a nonsense mode of 100000x100000 or something like that.
- OSD only works with double buffering enabled (else it flickers).

8.4. SDL

SDL (Simple Directmedia Layer) is basically a unified video/audio interface. Programs that use it know only about SDL, and not about what video or audio driver does SDL actually use. For example a Doom port using SDL can run on svgalib, aalib, X, fbdev, and others, you only have to specify the (for example) video driver to use with the `SDL_VIDEODRIVER` environment variable. Well, in theory.

With MPlayer, we used its X11 driver's software scaler ability for cards/drivers that doesn't support XVideo, until we made our own (faster, nicer) software scaler. Also we used its aalib output, but now we have ours which is more comfortable. Its DGA mode was better than ours, until recently. Get it now? :)

It also helps with some buggy drivers/cards if the video is jerky (not slow system problem), or audio is lagging.

SDL video output supports displaying subtitles under the movie, on the (if present) black bar.

8.5. SVGAlib

INSTALLATION. You'll have to install svgalib and its development package in order for MPlayer build its SVGAlib driver (autodetected, but can be forced), and don't forget to edit `/etc/vga/libvga.config` to suit your card and monitor.

Note

Be sure not to use the `-fs` switch, since it toggles the usage of the software scaler, and it's slow. If you really need it, use the `-sws 4` option which will produce bad quality, but is somewhat faster.

EGA (4BPP) SUPPORT. SVGAlib incorporates EGAlib, and MPlayer has the possibility to display any movie in 16 colors, thus usable in the following sets:

- EGA card with EGA monitor: 320x200x4bpp, 640x200x4bpp, 640x350x4bpp
- EGA card with CGA monitor: 320x200x4bpp, 640x200x4bpp

The bpp (bits per pixel) value must be set to 4 by hand: `-bpp 4`

The movie probably must be scaled down to fit in EGA mode:

```
-vf scale=640:350
```

or

```
-vf scale=320:200
```

For that we need fast but bad quality scaling routine:

```
-sws 4
```

Maybe automatic aspect correction has to be shut off:

```
-noaspect
```

Note

According to my experience the best image quality on EGA screens can be achieved by decreasing the brightness a bit: `-vf eq=-20:0`. I also needed to lower the audio samplerate on my box, because the sound was broken on 44kHz: `-srate 22050`.

You can turn on OSD and subtitles only with the `expand` filter, see the man page for exact parameters.

8.6. Framebuffer output (FBdev)

Whether to build the FBdev target is autodetected during `./configure`. Read the framebuffer documentation in the kernel sources (`Documentation/fb/*`) for more information.

If your card doesn't support VBE 2.0 standard (older ISA/PCI cards, such as S3 Trio64), only VBE 1.2 (or older?): Well, VESAfb is still available, but you'll have to load SciTech Display Doctor (formerly UniVBE) before booting Linux. Use a DOS boot disk or whatever. And don't forget to register your UniVBE ;))

The FBdev output takes some additional parameters above the others:

`-fb`

: specify the framebuffer device to use (default: `/dev/fb0`)

`-fbmode`

: mode name to use (according to `/etc/fb.modes`)

`-fbmodeconfig`

: config file of modes (default: `/etc/fb.modes`)

`-monitor-hfreq`, `-monitor-vfreq`, `-monitor-dotclock`

: **important** values, see `example.conf`

If you want to change to a specific mode, then use

```
mplayer -vm -fbmode name_of_mode filename
```

- `-vm` alone will choose the most suitable mode from `/etc/fb.modes`. Can be used together with `-x` and `-y` options too. The `-flip` option is supported only if the movie's pixel format matches the video mode's pixel format. Pay attention to the bpp value, fbdev driver tries to use the current, or if you specify the `-bpp` option, then that.
- `-zoom` option isn't supported (use `-vf scale`). You can't use 8bpp (or less) modes.
- You possibly want to turn the cursor off:

  ```
  echo -e '\033[?25l'
  ```

  or

  ```
  setterm -cursor off
  ```

  and the screen saver:

  ```
  setterm -blank 0
  ```

  To turn the cursor back on:

  ```
  echo -e '\033[?25h'
  ```

  or

  ```
  setterm -cursor on
  ```

Note

FBdev video mode changing *does not work* with the VESA framebuffer, and don't ask for it, since it's not an MPlayer limitation.

8.7. Matrox framebuffer (mga_vid)

`mga_vid` is a combination of a video output driver and a Linux kernel module that utilizes the Matrox G200/G400/G450/G550 video scaler/overlay unit to perform YUV->RGB colorspace conversion and arbitrary video scaling. `mga_vid` has hardware VSYNC support with triple buffering. It works on both a framebuffer console and under X, but only with Linux 2.4.x.

For a Linux 2.6.x version of this driver check out http://attila.kinali.ch/mga/ or have a look at the external Subversion repository of mga_vid which can be checked out via

```
svn checkout svn://svn.mplayerhq.hu/mga_vid
```

Installation:

1. To use it, you first have to compile `drivers/mga_vid.o`:

   ```
   make drivers
   ```

2. Then run (as `root`)

   ```
   make install-drivers
   ```

   which should install the module and create the device node for you. Load the driver with

   ```
   insmod mga_vid.o
   ```

3. You should verify the memory size detection using the **dmesg** command. If it's bad, use the `mga_ram_size` option (**rmmod mga_vid** first), specify card's memory size in MB:

   ```
   insmod mga_vid.o mga_ram_size=16
   ```

4. To make it load/unload automatically when needed, first insert the following line at the end of `/etc/modules.conf`:

   ```
   alias char-major-178 mga_vid
   ```

5. Now you have to (re)compile MPlayer, **./configure** will detect `/dev/mga_vid` and build the 'mga' driver. Using it from MPlayer goes by `-vo mga` if you have matroxfb console, or `-vo xmga` under XFree86 3.x.x or 4.x.x.

The mga_vid driver cooperates with Xv.

The `/dev/mga_vid` device file can be read for some info, for example by

```
cat /dev/mga_vid
```

and can be written for brightness change:

```
echo "brightness=120" > /dev/mga_vid
```

There is a test application called **mga_vid_test** in the same directory. It should draw 256x256 images on the screen if all is working well.

8.8. 3Dfx YUV support

This driver uses the kernel's tdfx framebuffer driver to play movies with YUV acceleration. You'll need a kernel with tdfxfb support, and recompile with

```
./configure --enable-tdfxfb
```

8.9. tdfx_vid

This is a combination of a Linux kernel module and a video output driver, similar to mga_vid. You'll need a 2.4.x kernel with the `agpgart` driver since `tdfx_vid` uses AGP. Pass `--enable-tdfxfb` to **configure** to build the video output driver and build the kernel module with the following instructions.

Installing the tdfx_vid.o kernel module:

1. Compile `drivers/tdfx_vid.o`:

   ```
   make drivers
   ```

2. Then run (as `root`)

   ```
   make install-drivers
   ```

   which should install the module and create the device node for you. Load the driver with

   ```
   insmod tdfx_vid.o
   ```

3. To make it load/unload automatically when needed, first insert the following line at the end of `/etc/modules.conf`:

   ```
   alias char-major-178 tdfx_vid
   ```

There is a test application called **tdfx_vid_test** in the same directory. It should print out some useful information if all is working well.

8.10. OpenGL output

MPlayer supports displaying movies using OpenGL, but if your platform/driver supports xv as should be the case on a PC with Linux, use xv instead, OpenGL performance is considerably worse. If you have an X11 implementation without xv support, OpenGL is a viable alternative.

Unfortunately not all drivers support this feature. The Utah-GLX drivers (for XFree86 3.3.6) support it for all cards. See http://utah-glx.sf.net for details about how to install it.

XFree86(DRI) 4.0.3 or later supports OpenGL with Matrox and Radeon cards, 4.2.0 or later supports Rage128. See http://dri.sf.net for download and installation instructions.

A hint from one of our users: the GL video output can be used to get vsynced TV output. You'll have to set an environment variable (at least on nVidia):

export __GL_SYNC_TO_VBLANK=1

8.11. AAlib – text mode displaying

AAlib is a library for displaying graphics in text mode, using powerful ASCII renderer. There are *lots* of programs already supporting it, like Doom, Quake, etc. MPlayer contains a very usable driver for it. If `./configure` detects aalib installed, the aalib libvo driver will be built.

You can use some keys in the AA Window to change rendering options:

Key	Action
1	decrease contrast
2	increase contrast
3	decrease brightness
4	increase brightness
5	switch fast rendering on/off
6	set dithering mode (none, error distribution, Floyd Steinberg)
7	invert image
8	toggles between aa and MPlayer control

The following command line options can be used:

`-aaosdcolor=V`

change OSD color

`-aasubcolor=V`

Change subtitle color

where *V* can be: `0` (normal), `1` (dark), `2` (bold), `3` (bold font), `4` (reverse), `5` (special).

AAlib itself provides a large sum of options. Here are some important:

`-aadriver`

Set recommended aa driver (X11, curses, Linux).

`-aaextended`

Use all 256 characters.

`-aaeight`

Use eight bit ASCII.

`-aahelp`

Prints out all aalib options.

Note

The rendering is very CPU intensive, especially when using AA-on-X (using aalib on X), and it's least CPU intensive on standard, non-framebuffer console. Use SVGATextMode to set up a big textmode, then enjoy! (secondary head Hercules cards rock :)) (but IMHO you can use `-vf 1bpp` option to get graphics on hgafb:)

Use the `-framedrop` option if your computer isn't fast enough to render all frames!

Playing on terminal you'll get better speed and quality using the Linux driver, not curses (`-aadriver linux`). But therefore you need write access on `/dev/vcsa<terminal>`! That isn't autodetected by aalib, but vo_aa tries to find the best mode. See http://aa-project.sf.net/tune for further tuning issues.

8.12. `libcaca` – Color ASCII Art library

The `libcaca` library is a graphics library that outputs text instead of pixels, so that it can work on older video cards or text terminals. It is not unlike the famous `AAlib` library. `libcaca` needs a terminal to work, thus it should work on all Unix systems (including Mac OS X) using either the `slang` library or the `ncurses` library, on DOS using the `conio.h` library, and on Windows systems using either `slang` or `ncurses` (through Cygwin emulation) or `conio.h`. If `./configure` detects `libcaca`, the caca libvo driver will be built.

The differences with `AAlib` are the following:

- 16 available colors for character output (256 color pairs)
- color image dithering

But `libcaca` also has the following limitations:

- no support for brightness, contrast, gamma

You can use some keys in the caca window to change rendering options:

Key	Action
d	Toggle `libcaca` dithering methods.
a	Toggle `libcaca` antialiasing.
b	Toggle `libcaca` background.

`libcaca` will also look for certain environment variables:

`CACA_DRIVER`

Set recommended caca driver. e.g. ncurses, slang, x11.

`CACA_GEOMETRY (X11 only)`

Specifies the number of rows and columns. e.g. 128x50.

`CACA_FONT (X11 only)`

Specifies the font to use. e.g. fixed, nexus.

Use the `-framedrop` option if your computer is not fast enough to render all frames.

8.13. VESA - output to VESA BIOS

This driver was designed and introduced as a **generic driver** for any video card which has VESA VBE 2.0 compatible BIOS. Another advantage of this driver is that it tries to force TV output on. *VESA BIOS EXTENSION (VBE) Version 3.0 Date: September 16, 1998* (Page 70) says:

> **Dual-Controller Designs.** VBE 3.0 supports the dual-controller design by assuming that since both controllers are typically provided by the same OEM, under control of a single BIOS ROM on the same graphics card, it is possible to hide the fact that two controllers are indeed present from the application. This has the limitation of preventing simultaneous use of the independent controllers, but allows applications released before VBE 3.0 to operate normally. The VBE Function 00h (Return Controller Information) returns the combined information of both controllers, including the combined list of available modes. When the application selects a mode, the appropriate controller is activated. Each of the remaining VBE functions then operates on the active controller.

So you have chances to get working TV-out by using this driver. (I guess that TV-out frequently is standalone head or standalone output at least.)

ADVANTAGES

- You have chances to watch movies **if Linux even doesn't know** your video hardware.

- You don't need to have installed any graphics' related things on your Linux (like X11 (AKA XFree86), fbdev and so on). This driver can be run from **text-mode**.
- You have chances to get **working TV-out**. (It's known at least for ATI's cards).
- This driver calls `int 10h` handler thus it's not an emulator - it calls **real** things of *real* BIOS in *real-mode* (actually in vm86 mode).
- You can use VIDIX with it, thus getting accelerated video display **and** TV output at the same time! (Recommended for ATI cards.)
- If you have VESA VBE 3.0+, and you had specified `monitor-hfreq, monitor-vfreq, monitor-dotclock` somewhere (config file, or command line) you will get the highest possible refresh rate. (Using General Timing Formula). To enable this feature you have to specify **all** your monitor options.

DISADVANTAGES

- It works only on **x86 systems**.
- It can be used only by `root`.
- Currently it's available only for **Linux**.

Important

Don't use this driver with **GCC 2.96**! It won't work!

COMMAND LINE OPTIONS AVAILABLE FOR VESA

`-vo vesa:`*`opts`*

currently recognized: `dga` to force dga mode and `nodga` to disable dga mode. In dga mode you can enable double buffering via the `-double` option. Note: you may omit these parameters to enable **autodetection** of dga mode.

KNOWN PROBLEMS AND WORKAROUNDS

- If you have installed **NLS** font on your Linux box and run VESA driver from text-mode then after terminating MPlayer you will have **ROM font** loaded instead of national. You can load national font again by using **setsysfont** utility from the Mandrake/Mandriva distribution for example. (**Hint**: The same utility is used for localization of fbdev).
- Some **Linux graphics drivers** don't update active **BIOS mode** in DOS memory. So if you have such problem - always use VESA driver only from **text-mode**. Otherwise text-mode (#03) will be activated anyway and you will need restart your computer.
- Often after terminating VESA driver you get **black** screen. To return your screen to original state - simply switch to other console (by pressing **Alt**+**F<x>**) then switch to your previous console by the same way.
- To get **working TV-out** you need have plugged TV-connector in before booting your PC since video BIOS initializes itself only once during POST procedure.

8.14. X11

Avoid if possible. Outputs to X11 (uses shared memory extension), with no hardware acceleration at all. Supports (MMX/3DNow/SSE accelerated, but still slow) software scaling, use the options `-fs -zoom`. Most cards have hardware scaling support, use the `-vo xv` output for them, or `-vo xmga` for Matrox cards.

The problem is that most cards' driver doesn't support hardware acceleration on the second head/TV. In those cases, you see green/blue colored window instead of the movie. This is where this driver comes in handy, but you need powerful CPU to use software scaling. Don't use the SDL driver's software output+scaler, it has worse image quality!

Software scaling is very slow, you better try changing video modes instead. It's very simple. See the DGA section's modelines, and insert them into your `XF86Config`.

- If you have XFree86 4.x.x: use the `-vm` option. It will change to a resolution your movie fits in. If it doesn't:
- With XFree86 3.x.x: you have to cycle through available resolutions with the **Ctrl**+**Alt**+**Keypad +** and **Ctrl**+**Alt**+**Keypad -** keys.

If you can't find the modes you inserted, browse XFree86's output. Some drivers can't use low pixelclocks that are needed for low resolution video modes.

8.15. VIDIX

PREAMBLE. VIDIX is the abbreviation for **VID**eo **I**nterface for *ni**X**. VIDIX was designed and introduced as an interface for fast user-space drivers providing such video performance as mga_vid does for Matrox cards. It's also very portable.

This interface was designed as an attempt to fit existing video acceleration interfaces (known as mga_vid, rage128_vid, radeon_vid, pm3_vid) into a fixed scheme. It provides a high level interface to chips which are known as BES (BackEnd scalers) or OV (Video Overlays). It doesn't provide low level interface to things which are known as graphics servers. (I don't want to compete with X11 team in graphics mode switching). I.e. main goal of this interface is to maximize the speed of video playback.

USAGE

- You can use standalone video output driver: `-vo xvidix`. This driver was developed as X11's front end to VIDIX technology. It requires X server and can work only under X server. Note that, as it directly accesses the hardware and circumvents the X driver, pixmaps cached in the graphics card's memory may be corrupted. You can prevent this by limiting the amount of video memory used by X with the XF86Config option "VideoRam" in the device section. You should set this to the amount of memory installed on your card minus 4MB. If you have less than 8MB of video ram, you can use the option "XaaNoPixmapCache" in the screen section instead.
- There is a console VIDIX driver: `-vo cvidix`. This requires a working and initialized framebuffer for most cards (or else you'll just mess up the screen), and you'll have a similar effect as with `-vo mga` or `-vo fbdev`. nVidia cards however are able to output truly graphical video on a real text console. See the nvidia_vid section for more information. To get rid of text on the borders and the blinking cursor, try something like

  ```
  setterm -cursor off > /dev/tty9
  ```

  (assuming `tty9` is unused so far) and then switch to `tty9`. On the other hand, `-colorkey 0` should give you a video running in the "background", though this depends on the colorkey functionality to work right.
- You can use VIDIX subdevice which was applied to several video output drivers, such as: `-vo vesa:vidix` (**Linux only**) and `-vo fbdev:vidix`.

Indeed it doesn't matter which video output driver is used with **VIDIX**.

REQUIREMENTS

- Video card should be in graphics mode (except nVidia cards with the `-vo cvidix` output driver).
- MPlayer's video output driver should know active video mode and be able to tell to VIDIX subdevice some video characteristics of server.

USAGE METHODS. When VIDIX is used as **subdevice** (`-vo vesa:vidix`) then video mode configuration is performed by video output device (**vo_server** in short). Therefore you can pass into command line of MPlayer the same keys as for vo_server. In addition it understands `-double` key as globally visible parameter. (I recommend using this key with VIDIX at least for ATI's card). As for `-vo xvidix`, currently it recognizes the following options: `-fs -zoom -x -y -double`.

Also you can specify VIDIX's driver directly as third subargument in command line:

```
mplayer -vo xvidix:mga_vid.so -fs -zoom -double file.avi
```

or

```
mplayer -vo vesa:vidix:radeon_vid.so -fs -zoom -double -bpp 32 file.avi
```

But it’s dangerous, and you shouldn’t do that. In this case given driver will be forced and result is unpredictable (it may **freeze** your computer). You should do that ONLY if you are absolutely sure it will work, and MPlayer doesn’t do it automatically. Please tell about it to the developers. The right way is to use VIDIX without arguments to enable driver autodetection.

8.15.1. svgalib_helper

Since VIDIX requires direct hardware access you can either run it as root or set the SUID bit on the MPlayer binary (**Warning: This is a security risk!**). Alternatively, you can use a special kernel module, like this:

1. Download the development version of svgalib (1.9.x).
2. Compile the module in the `svgalib_helper` directory (it can be found inside the `svgalib-1.9.17/kernel/` directory if you’ve downloaded the source from the svgalib site) and insmod it.
3. To create the necessary devices in the `/dev` directory, do a

   ```
   make device
   ```

   in the `svgalib_helper` dir, as root.
4. Then run `configure` again and pass the parameter `--enable-svgalib_helper` as well as `--with-extraincdir=/path/to/svgalib_helper/sources/kernel/svgalib_helper`, where `/path/to/svgalib_helper/sources/` has to be adjusted to wherever you extracted svgalib_helper sources.
5. Recompile.

8.15.2. ATI cards

Currently most ATI cards are supported natively, from Mach64 to the newest Radeons.

There are two compiled binaries: `radeon_vid` for Radeon and `rage128_vid` for Rage 128 cards. You may force one or let the VIDIX system autoprobe all available drivers.

8.15.3. Matrox cards

Matrox G200, G400, G450 and G550 have been reported to work.

The driver supports video equalizers and should be nearly as fast as the Matrox framebuffer

8.15.4. Trident cards

There is a driver available for the Trident Cyberblade/i1 chipset, which can be found on VIA Epia motherboards.

The driver was written and is maintained by Alastair M. Robinson.

8.15.5. 3DLabs cards

Although there is a driver for the 3DLabs GLINT R3 and Permedia3 chips, no one has tested it, so reports are welcome.

8.15.6. nVidia cards

An unique feature of the nvidia_vid driver is its ability to display video on **plain, pure, text-only console** - with no framebuffer or X magic whatsoever. For this purpose, we'll have to use the `cvidix` video output, as the following example shows:

```
mplayer -vo cvidix example.avi
```

8.15.7. SiS cards

This is very experimental code, just like nvidia_vid.

It's been tested on SiS 650/651/740 (the most common chipsets used in the SiS versions of the "Shuttle XPC" barebones boxes out there)

Reports awaited!

8.16. DirectFB

> "DirectFB is a graphics library which was designed with embedded systems in mind. It offers maximum hardware accelerated performance at a minimum of resource usage and overhead." - quoted from http://www.directfb.org

I'll exclude DirectFB features from this section.

Though MPlayer is not supported as a "video provider" in DirectFB, this output driver will enable video playback through DirectFB. It will - of course - be accelerated, on my Matrox G400 DirectFB's speed was the same as XVideo.

Always try to use the newest version of DirectFB. You can use DirectFB options on the command line, using the `-dfbopts` option. Layer selection can be done by the subdevice method, e.g.: `-vo directfb:2` (layer -1 is default: autodetect)

8.17. DirectFB/Matrox (dfbmga)

Please read the main DirectFB section for general information.

This video output driver will enable CRTC2 (on the second head) on Matrox G400/G450/G550 cards, displaying video **independent** of the first head.

Ville Syrjala's has a README and a HOWTO on his homepage that explain how to make DirectFB TV output run on Matrox cards.

Note

the first DirectFB version with which we could get this working was 0.9.17 (it's buggy, needs that `surfacemanager` patch from the URL above). Porting the CRTC2 code to mga_vid has been planned for years, patches are welcome.

8.18. MPEG decoders

8.18.1. DVB output and input

MPlayer supports cards with the Siemens DVB chipset from vendors like Siemens, Technotrend, Galaxis or Hauppauge. The latest DVB drivers are available from the Linux TV site. If you want to do software transcoding you should have at least a 1GHz CPU.

Configure should detect your DVB card. If it did not, force detection with

```
./configure --enable-dvb
```

If you have ost headers at a non-standard path, set the path with

```
./configure --with-extraincdir=DVB source directory/ost/include
```

Then compile and install as usual.

USAGE. Hardware decoding of streams containing MPEG-1/2 video and/or MPEG audio can be done with this command:

```
mplayer -ao mpegpes -vo mpegpes file.mpg|vob
```

Decoding of any other type of video stream requires transcoding to MPEG-1, thus it's slow and may not be worth the trouble, especially if your computer is slow. It can be achieved using a command like this:

```
mplayer -ao mpegpes -vo mpegpes yourfile.ext
mplayer -ao mpegpes -vo mpegpes -vf expand yourfile.ext
```

Note that DVB cards only support heights 288 and 576 for PAL or 240 and 480 for NTSC. You **must** rescale for other heights by adding `scale=width:height` with the width and height you want to the `-vf` option. DVB cards accept various widths, like 720, 704, 640, 512, 480, 352 etc. and do hardware scaling in horizontal direction, so you do not need to scale horizontally in most cases. For a 512x384 (aspect 4:3) MPEG-4 (DivX) try:

```
mplayer -ao mpegpes -vo mpegpes -vf scale=512:576
```

If you have a widescreen movie and you do not want to scale it to full height, you can use the `expand=w:h` filter to add black bands. To view a 640x384 MPEG-4 (DivX), try:

```
mplayer -ao mpegpes -vo mpegpes -vf expand=640:576 file.avi
```

If your CPU is too slow for a full size 720x576 MPEG-4 (DivX), try downscaling:

```
mplayer -ao mpegpes -vo mpegpes -vf scale=352:576 file.avi
```

If speed does not improve, try vertical downscaling, too:

```
mplayer -ao mpegpes -vo mpegpes -vf scale=352:288 file.avi
```

For OSD and subtitles use the OSD feature of the expand filter. So, instead of `expand=w:h` or `expand=w:h:x:y`, use `expand=w:h:x:y:1` (the 5th parameter `:1` at the end will enable OSD rendering). You may want to move the image up a bit to get a bigger black zone for subtitles. You may also want to move subtitles up, if they are outside your TV screen, use the `-subpos <0-100>` option to adjust this (`-subpos 80` is a good choice).

In order to play non-25fps movies on a PAL TV or with a slow CPU, add the `-framedrop` option.

To keep the aspect ratio of MPEG-4 (DivX) files and get the optimal scaling parameters (hardware horizontal scaling and software vertical scaling while keeping the right aspect ratio), use the new dvbscale filter:

```
for a  4:3 TV: -vf dvbscale,scale=-1:0,expand=-1:576:-1:-1:1
for a 16:9 TV: -vf dvbscale=1024,scale=-1:0,expand=-1:576:-1:-1:1
```

Digital TV (DVB input module). You can use your DVB card for watching Digital TV.

You should have the programs **scan** and **szap/tzap/czap/azap** installed; they are all included in the drivers package.

Verify that your drivers are working properly with a program such as **dvbstream** (that is the base of the DVB input module).

Now you should compile a `~/.mplayer/channels.conf` file, with the syntax accepted by **szap/tzap/czap/azap**, or have

scan compile it for you.

If you have more than one card type (e.g. Satellite, Terrestrial, Cable and ATSC) you can save your channels files as `~/.mplayer/channels.conf.sat`, `~/.mplayer/channels.conf.ter`, `~/.mplayer/channels.conf.cbl`, and `~/.mplayer/channels.conf.atsc`, respectively, so as to implicitly hint MPlayer to use these files rather than `~/.mplayer/channels.conf`, and you only need to specify which card to use.

Make sure that you have *only* Free to Air channels in your `channels.conf` file, or MPlayer will wait for an unencrypted transmission.

In your audio and video fields you can use an extended syntax: `...:pid[+pid]:...` (for a maximum of 6 pids each); in this case MPlayer will include in the stream all the indicated pids, plus pid 0 (that contains the PAT). You should always include in each row the PMT and PCR pids for the corresponding channel (if you know them). You can also specify 8192, this will select all pids on this frequency and you can then switch between the programs with TAB. This might need more bandwidth, though cheap cards always transfer all channels at least to the kernel so it does not make much of a difference for these. Other possible uses are: televideo pid, second audio track, etc.

If MPlayer complains frequently about

```
Too many video/audio packets in the buffer
```

or if you notice a growing desynchronization between audio and video verify the presence of the PCR pid in your stream (needed to comply with the buffering model of the transmitter) and/or try to use the libavformat MPEG-TS demuxer by adding `-demuxer lavf -lavfdopts probesize=128` to your command line.

To show the first of the channels present in your list, run

```
mplayer dvb://
```

If you want to watch a specific channel, such as R1, run

```
mplayer dvb://R1
```

If you have more than one card you also need to specify the number of the card where the channel is visible (e.g. 2) with the syntax:

```
mplayer dvb://2@R1
```

To change channels press the **h** (next) and **k** (previous) keys, or use the OSD menu.

To temporarily disable the audio or the video stream copy the following to `~/.mplayer/input.conf`:

```
% set_property  switch_video -2
& step_property switch_video
? set_property  switch_audio -2
^ step_property switch_audio
```

(Overriding the action keys as preferred.) When pressing the key corresponding to switch_x -2 the associated stream will be closed; when pressing the key corresponding to step_x the stream will be reopened. Notice that this switching mechanism will not work as expected when there are multiple audio and video streams in the multiplex.

During playback (not during recording), in order to prevent stuttering and error messages such as 'Your system is too slow' it is advisable to add

```
-mc 10 -speed 0.97 -af scaletempo
```

to your command line, adjusting the scaletempo parameters as preferred.

If your `~/.mplayer/menu.conf` contains a `<dvbsel>` entry, such as the one in the example file `etc/dvb-menu.conf` (that you can use to overwrite `~/.mplayer/menu.conf`), the main menu will show a sub-menu entry that will permit you to choose one of the channels present in your `channels.conf`, possibly preceded by a menu with the list of cards available if more than one is usable by MPlayer.

If you want to save a program to disk you can use

```
mplayer -dumpfile r1.ts -dumpstream dvb://R1
```

If you want to record it in a different format (re-encoding it) instead you can run a command such as

```
mencoder -o r1.avi -ovc xvid -xvidencopts bitrate=800 \
    -oac mp3lame -lameopts cbr:br=128 -pp=ci dvb://R1
```

Read the man page for a list of options that you can pass to the DVB input module.

FUTURE. If you have questions or want to hear feature announcements and take part in discussions on this subject, join our MPlayer-DVB mailing list. Please remember that the list language is English.

In the future you may expect the ability to display OSD and subtitles using the native OSD feature of DVB cards.

8.18.2. DXR2

MPlayer supports hardware accelerated playback with the Creative DXR2 card.

First of all you will need properly installed DXR2 drivers. You can find the drivers and installation instructions at the DXR2 Resource Center site.

USAGE

`-vo dxr2`

Enable TV output.

`-vo dxr2:x11` or `-vo dxr2:xv`

Enable Overlay output in X11.

`-dxr2 <option1:option2:...>`

This option is used to control the DXR2 driver.

The overlay chipset used on the DXR2 is of pretty bad quality but the default settings should work for everybody. The OSD may be usable with the overlay (not on TV) by drawing it in the colorkey. With the default colorkey settings you may get variable results, usually you will see the colorkey around the characters or some other funny effect. But if you properly adjust the colorkey settings you should be able to get acceptable results.

Please see the man page for available options.

8.18.3. DXR3/Hollywood+

MPlayer supports hardware accelerated playback with the Creative DXR3 and Sigma Designs Hollywood Plus cards. These cards both use the em8300 MPEG decoder chip from Sigma Designs.

First of all you will need properly installed DXR3/H+ drivers, version 0.12.0 or later. You can find the drivers and installation instructions at the DXR3 & Hollywood Plus for Linux site. `configure` should detect your card automatically, compilation should go without problems.

USAGE

`-vo dxr3:prebuf:sync:norm=x:device`

`overlay` activates the overlay instead of TV-out. It requires that you have a properly configured overlay setup to work right. The easiest way to configure the overlay is to first run autocal. Then run mplayer with dxr3 output and without overlay turned on, run dxr3view. In dxr3view you can tweak the overlay settings and see the effects in realtime, perhaps this feature will be supported by the MPlayer GUI in the future. When overlay is properly set up you will no longer need to use dxr3view. `prebuf` turns on prebuffering. Prebuffering is a feature of the em8300 chip that enables it to hold more than one frame of video at a time. This means that when you are running with prebuffering MPlayer will try to keep the video buffer filled with data at all times. If you are on a slow machine MPlayer will probably use close to, or precisely 100% of CPU. This is especially common if you play pure MPEG streams (like DVDs, SVCDs a.s.o.) since MPlayer will not have to reencode it to MPEG it will fill the buffer very fast. With prebuffering video playback is **much** less sensitive to other programs hogging the CPU, it will not drop frames unless applications hog the CPU for a long time. When running without prebuffering the em8300 is much more sensitive to CPU load, so it is highly suggested that you turn on MPlayer's `-framedrop` option to avoid further loss of sync. `sync` will turn on the new sync-engine. This is currently an experimental feature. With the sync feature turned on the em8300's internal clock will be monitored at all times, if it starts to deviate from MPlayer's clock it will be reset causing the em8300 to drop any frames that are lagging behind. `norm=x` will set the TV norm of the DXR3 card without the need for external tools like em8300setup. Valid norms are 5 = NTSC, 4 = PAL-60, 3 = PAL. Special norms are 2 (auto-adjust using PAL/PAL-60) and 1 (auto-adjust using PAL/NTSC) because they decide which norm to use by looking at the frame rate of the movie. norm = 0 (default) does not change the current norm. *`device`* = device number to use if you have more than one em8300 card. Any of these options may be left out. `:prebuf:sync` seems to work great when playing MPEG-4 (DivX) movies. People have reported problems using the prebuf option when playing MPEG-1/2 files. You might want to try running without any options first, if you have sync problems, or DVD subtitle problems, give `:sync` a try.

`-ao oss:/dev/em8300_ma-`*`X`*

For audio output, where *X* is the device number (0 if one card).

`-af resample=`*`xxxxx`*

The em8300 cannot play back samplerates lower than 44100Hz. If the sample rate is below 44100Hz select either 44100Hz or 48000Hz depending on which one matches closest. I.e. if the movie uses 22050Hz use 44100Hz as 44100 / 2 = 22050, if it is 24000Hz use 48000Hz as 48000 / 2 = 24000 and so on. This does not work with digital audio output (`-ac hwac3`).

`-vf lavc`

To watch non-MPEG content on the em8300 (i.e. MPEG-4 (DivX) or RealVideo) you have to specify an MPEG-1 video filter such as `libavcodec` (lavc). See the man page for further info about `-vf lavc`. Currently there is no way of setting the fps of the em8300 which means that it is fixed to 30000/1001 fps. Because of this it is highly recommended that you use `-vf lavc=`*`quality`*`:25` especially if you are using prebuffering. Then why 25 and not 30000/1001? Well, the thing is that when you use 30000/1001 the picture becomes a bit jumpy. The reason for this is unknown to us. If you set it to somewhere between 25 and 27 the picture becomes stable. For now all we can do is accept this for a fact.

`-vf expand=-1:-1:-1:-1:1`

Although the DXR3 driver can put some OSD onto the MPEG-1/2/4 video, it has much lower quality than MPlayer's traditional OSD, and has several refresh problems as well. The command line above will firstly convert the input video to MPEG-4 (this is mandatory, sorry), then apply an expand filter which won't expand anything (-1: default), but apply the normal OSD onto the picture (that's what the "1" at the end does).

`-ac hwac3`

The em8300 supports playing back AC-3 audio (surround sound) through the digital audio output of the card. See the `-ao oss` option above, it must be used to specify the DXR3's output instead of a sound card.

8.19. Other visualization hardware

8.19.1. Zr

This is a display-driver (`-vo zr`) for a number of MJPEG capture/playback cards (tested for DC10+ and Buz, and it should work for the LML33, the DC10). The driver works by encoding the frame to JPEG and then sending it to the card. For the

JPEG encoding `libavcodec` is used, and required. With the special *cinerama* mode, you can watch movies in true wide screen provided that you have two beamers and two MJPEG cards. Depending on resolution and quality settings, this driver may require a lot of CPU power, remember to specify `-framedrop` if your machine is too slow. Note: My AMD K6-2 350MHz is (with `-framedrop`) quite adequate for watching VCD sized material and downscaled movies.

This driver talks to the kernel driver available at http://mjpeg.sf.net, so you must get it working first. The presence of an MJPEG card is autodetected by the `configure` script, if autodetection fails, force detection with

```
./configure --enable-zr
```

The output can be controlled by several options, a long description of the options can be found in the man page, a short list of options can be viewed by running

```
mplayer -zrhelp
```

Things like scaling and the OSD (on screen display) are not handled by this driver but can be done using the video filters. For example, suppose that you have a movie with a resolution of 512x272 and you want to view it fullscreen on your DC10+. There are three main possibilities, you may scale the movie to a width of 768, 384 or 192. For performance and quality reasons, I would choose to scale the movie to 384x204 using the fast bilinear software scaler. The command line is

```
mplayer -vo zr -sws 0 -vf scale=384:204 movie.avi
```

Cropping can be done by the `crop` filter and by this driver itself. Suppose that a movie is too wide for display on your Buz and that you want to use `-zrcrop` to make the movie less wide, then you would issue the following command

```
mplayer -vo zr -zrcrop 720x320+80+0 benhur.avi
```

if you want to use the `crop` filter, you would do

```
mplayer -vo zr -vf crop=720:320:80:0 benhur.avi
```

Extra occurrences of `-zrcrop` invoke *cinerama* mode, i.e. you can distribute the movie over several TVs or beamers to create a larger screen. Suppose you have two beamers. The left one is connected to your Buz at `/dev/video1` and the right one is connected to your DC10+ at `/dev/video0`. The movie has a resolution of 704x288. Suppose also that you want the right beamer in black and white and that the left beamer should have JPEG frames at quality 10, then you would issue the following command

```
mplayer -vo zr -zrdev /dev/video0 -zrcrop 352x288+352+0 -zrxdoff 0 -zrbw \
    -zrcrop 352x288+0+0 -zrdev /dev/video1 -zrquality 10 \
        movie.avi
```

You see that the options appearing before the second `-zrcrop` only apply to the DC10+ and that the options after the second `-zrcrop` apply to the Buz. The maximum number of MJPEG cards participating in *cinerama* is four, so you can build a 2x2 vidiwall.

Finally an important remark: Do not start or stop XawTV on the playback device during playback, it will crash your computer. It is, however, fine to **FIRST** start XawTV, **THEN** start MPlayer, wait for MPlayer to finish and **THEN** stop XawTV.

8.19.2. Blinkenlights

This driver is capable of playback using the Blinkenlights UDP protocol. If you don't know what Blinkenlights or its successor Arcade are, find it out. Although this is most probably the least used video output driver, without a doubt it is the coolest MPlayer has to offer. Just watch some of the Blinkenlights documentation videos. On the Arcade video you can see the Blinkenlights output driver in action at 00:07:50.

8.20. TV-out support

8.20.1. Matrox G400 cards

Under Linux you have two methods to get G400 TV out working:

Important

for Matrox G450/G550 TV-out instructions, please see the next section!

XFree86

Using the driver and the HAL module, available from the Matrox site. This will give you X on the TV.

This method doesn't give you accelerated playback as under Windows! The second head has only YUV framebuffer, the *BES* (Back End Scaler, the YUV scaler on G200/G400/G450/G550 cards) doesn't work on it! The windows driver somehow workarounds this, probably by using the 3D engine to zoom, and the YUV framebuffer to display the zoomed image. If you really want to use X, use the `-vo x11 -fs -zoom` options, but it will be **SLOW**, and has **Macrovision** copy protection enabled (you can "workaround" Macrovision using this Perl script).

Framebuffer

Using the **matroxfb modules** in the 2.4 kernels. 2.2 kernels don't have the TV-out feature in them, thus unusable for this. You have to enable ALL matroxfb-specific features during compilation (except MultiHead), and compile them into **modules**! You'll also need to enable I2C and put the tools matroxset, fbset and con2fb in your path.

1. Then load the `matroxfb_Ti3026, matroxfb_maven, i2c-matroxfb, matroxfb_crtc2` modules into your kernel. Your text-mode console will enter into framebuffer mode (no way back!).

2. Next, set up your monitor and TV to your liking using the above tools.

3. Yoh. Next task is to make the cursor on tty1 (or whatever) to disappear, and turn off screen blanking. Execute the following commands:

   ```
   echo -e '\033[?25l'
   setterm -blank 0
   ```

   or

   ```
   setterm -cursor off
   setterm -blank 0
   ```

   You possibly want to put the above into a script, and also clear the screen. To turn the cursor back:

   ```
   echo -e '\033[?25h'
   ```

   or

   ```
   setterm -cursor on
   ```

4. Yeah kewl. Start movie playing with

   ```
   mplayer -vo mga -fs -screenw 640 -screenh 512 filename
   ```

   (If you use X, now change to matroxfb with for example **Ctrl**+**Alt**+**F1**.) Change `640` and `512` if you set the resolution to other...

5. **Enjoy the ultra-fast ultra-featured Matrox TV output (better than Xv)!**

8.20.2. Matrox G450/G550 cards

TV output support for these cards has only been recently introduced, and is not yet in the mainstream kernel. Currently the **mga_vid** module can't be used AFAIK, because the G450/G550 driver works only in one configuration: the first CRTC chip (with much more features) on the first display (on monitor), and the second CRTC (no **BES** - for explanation on BES, please see the G400 section above) on TV. So you can only use MPlayer's *fbdev* output driver at the present.

The first CRTC can't be routed to the second head currently. The author of the kernel matroxfb driver - Petr Vandrovec - will maybe make support for this, by displaying the first CRTC's output onto both of the heads at once, as currently recommended for G400, see the section above.

The necessary kernel patch and the detailed HOWTO is downloadable from http://www.bglug.ca/matrox_tvout/

8.20.3. Building a Matrox TV-out cable

No one takes any responsibility, nor guarantee for any damage caused by this documentation.

Cable for G400. The CRTC2 connector's fourth pin is the composite video signal. The ground are the sixth, seventh and eighth pins. (info contributed from Balázs Rácz)

Cable for G450. The CRTC2 connector's first pin is the composite video signal. The ground are the fifth, sixth, seventh, and fifteenth (5, 6, 7, 15) pins. (info contributed from Balázs Kerekes)

8.20.4. ATI cards

PREAMBLE. Currently ATI doesn't want to support any of its TV-out chips under Linux, because of their licensed Macrovision technology.

ATI CARDS TV-OUT STATUS ON LINUX

- **ATI Mach64**: supported by GATOS.
- **ASIC Radeon VIVO**: supported by GATOS.
- **Radeon** and **Rage128**: supported by MPlayer! Check VESA driver and VIDIX sections.
- **Rage Mobility P/M, Radeon, Rage 128, Mobility M3/M4**: supported by atitvout.

On other cards, just use the VESA driver, without VIDIX. Powerful CPU is needed, though.

Only thing you need to do - **Have the TV connector plugged in before booting your PC** since video BIOS initializes itself only once during POST procedure.

8.20.5. nVidia

First, you MUST download the closed-source drivers from http://nvidia.com. I will not describe the installation and configuration process because it does not cover the scope of this documentation.

After XFree86, XVideo, and 3D acceleration is properly working, edit your card's Device section in the `XF86Config` file, according to the following example (adapt for your card/TV):

```
Section "Device"
        Identifier      "GeForce"
        VendorName      "ASUS"
        BoardName       "nVidia GeForce2/MX 400"
        Driver          "nvidia"
        #Option         "NvAGP" "1"
        Option          "NoLogo"
        Option          "CursorShadow"  "on"

        Option          "TwinView"
        Option          "TwinViewOrientation" "Clone"
        Option          "MetaModes" "1024x768,640x480"
        Option          "ConnectedMonitor" "CRT, TV"
        Option          "TVStandard" "PAL-B"
        Option          "TVOutFormat" "Composite"
EndSection
```

Of course the important thing is the TwinView part.

8.20.6. NeoMagic

The NeoMagic chip is found in a variety of laptops, some of them are equipped with a simple analog TV encoder, some have a more advanced one.

- **Analog encoder chip**: It has been reported that reliable TV out can be obtained by using `-vo fbdev` or `-vo fbdev2`. You need to have vesafb compiled in your kernel and pass the following parameters on the kernel command line: `append="video=vesafb:ywrap,mtrr" vga=791`. You should start X, then switch to console mode with e.g. **Ctrl**+**Alt**+**F1**. If you fail to start X before running MPlayer from the console, the video becomes slow and choppy (explanations are welcome). Login to your console, then initiate the following command:

  ```
  clear; mplayer -vo fbdev -zoom -cache 8192 dvd://
  ```

  Now you should see the movie running in console mode filling up about half your laptop's LCD screen. To switch to TV hit **Fn**+**F5** three times. Tested on a Tecra 8000, 2.6.15 kernel with vesafb, ALSA v1.0.10.

- **Chrontel 70xx encoder chip**: Found in IBM Thinkpad 390E and possibly other Thinkpads or notebooks.

  You must use `-vo vesa:neotv_pal` for PAL or `-vo vesa:neotv_ntsc` for NTSC. It will provide TV output function in the following 16 bpp and 8 bpp modes:

  - NTSC 320x240, 640x480 and maybe 800x600 too.
  - PAL 320x240, 400x300, 640x480, 800x600.

  Mode 512x384 is not supported in BIOS. You must scale the image to a different resolution to activate TV out. If you can see an image on the screen in 640x480 or in 800x600 but not in 320x240 or other smaller resolution you need to replace two tables in `vbelib.c`. See the vbeSetTV function for details. Please contact the author in this case.

  Known issues: VESA-only, no other controls such as brightness, contrast, blacklevel, flickfilter are implemented.

Chapter 9. Ports

Binary packages of MPlayer are available from several sources. We have a list of places to get unofficial packages for various systems on our homepage. However, **none of these packages are supported**. Report problems to the authors, not to us.

9.1. Linux

9.1.1. Debian packaging

To build a Debian package, run the following command in the MPlayer source directory:

```
fakeroot debian/rules binary
```

If you want to pass custom options to configure, you can set up the `DEB_BUILD_OPTIONS` environment variable. For instance, if you want GUI and OSD menu support you would use:

```
DEB_BUILD_OPTIONS="--enable-gui --enable-menu" fakeroot debian/rules binary
```

You can also pass some variables to the Makefile. For example, if you want to compile with gcc 3.4 even if it's not the default compiler:

```
CC=gcc-3.4 DEB_BUILD_OPTIONS="--enable-gui" fakeroot debian/rules binary
```

To clean up the source tree run the following command:

```
fakeroot debian/rules clean
```

As root you can then install the `.deb` package as usual:

```
dpkg -i ../mplayer_version.deb
```

9.1.2. RPM packaging

To build an RPM package, run the following command in the MPlayer source directory:

```
FIXME: insert proper commands here
```

9.1.3. ARM Linux

MPlayer works on Linux PDAs with ARM CPU e.g. Sharp Zaurus, Compaq Ipaq. The easiest way to obtain MPlayer is to get it from one of the OpenZaurus package feeds. If you want to compile it yourself, you should look at the mplayer and the libavcodec directory in the OpenZaurus distribution buildroot. These always have the latest Makefile and patches used for building a SVN MPlayer. If you need a GUI frontend, you can use xmms-embedded.

9.2. *BSD

MPlayer runs on all known BSD flavors. There are ports/pkgsrc/fink/etc versions of MPlayer available that are probably easier to use than our raw sources.

If MPlayer complains about not finding `/dev/cdrom` or `/dev/dvd`, create an appropriate symbolic link:

```
ln -s /dev/your_cdrom_device /dev/cdrom
```

To use Win32 DLLs with MPlayer you will need to re-compile the kernel with "`option USER_LDT`" (unless you run FreeBSD-CURRENT, where this is the default).

9.2.1. FreeBSD

If your CPU has SSE, recompile your kernel with "`options CPU_ENABLE_SSE`" (FreeBSD-STABLE or kernel patches required).

9.2.2. OpenBSD

Due to limitations in different versions of gas (relocation vs MMX), you will need to compile in two steps: First make sure that the non-native as is first in your `$PATH` and do a **gmake -k**, then make sure that the native version is used and do **gmake**.

As of OpenBSD 3.4 the hack above is no longer needed.

9.2.3. Darwin

See the Mac OS section.

9.3. Commercial Unix

MPlayer has been ported to a number of commercial Unix variants. Since the development environments on these systems tend to be different from those found on free Unixes, you may have to make some manual adjustments to make the build work.

9.3.1. Solaris

Solaris still has broken, POSIX-incompatible system tools and shell in default locations. Until a bold step out of the computing stone age is made, you will have to add `/usr/xpg4/bin` to your `PATH`.

MPlayer should work on Solaris 2.6 or newer. Use the SUN audio driver with the `-ao sun` option for sound.

On **UltraSPARCs**, MPlayer takes advantage of their **VIS** extensions (equivalent to MMX), currently only in `libmpeg2`, `libvo` and `libavcodec`, but not in `mp3lib`. You can watch a VOB file on a 400MHz CPU. You'll need `mLib` installed.

Caveat:

- **mediaLib** is **currently disabled** by default in MPlayer because of brokenness. SPARC users who build MPlayer with mediaLib support have reported a thick, green-tint on video encoded and decoded with libavcodec. You may enable it if you wish with:

  ```
  ./configure --enable-mlib
  ```

  You do this at your own risk. x86 users should **never** use mediaLib, as this will result in very poor MPlayer performance.

On Solaris SPARC, you need the GNU C/C++ Compiler; it does not matter if GNU C/C++ compiler is configured with or without the GNU assembler.

On Solaris x86, you need the GNU assembler and the GNU C/C++ compiler, configured to use the GNU assembler! The MPlayer code on the x86 platform makes heavy use of MMX, SSE and 3DNOW! instructions that cannot be compiled using Sun's assembler `/usr/ccs/bin/as`.

The `configure` script tries to find out, which assembler program is used by your "gcc" command (in case the autodetection fails, use the `--as=/wherever/you/have/installed/gnu-as` option to tell the `configure` script where it can find GNU "as" on your system).

Solutions to common problems:

- Error message from `configure` on a Solaris x86 system using GCC without GNU assembler:

  ```
  % configure
  ```

```
...
Checking assembler (/usr/ccs/bin/as) ... , failed
Please upgrade(downgrade) binutils to 2.10.1...
```

(Solution: Install and use a gcc configured with `--with-as=gas`)

Typical error you get when building with a GNU C compiler that does not use GNU as:

```
% gmake
...
gcc -c -Iloader -Ilibvo -O4 -march=i686 -mcpu=i686 -pipe -ffast-math
     -fomit-frame-pointer  -I/usr/local/include   -o mplayer.o mplayer.c
Assembler: mplayer.c
"(stdin)", line 3567 : Illegal mnemonic
"(stdin)", line 3567 : Syntax error
... more "Illegal mnemonic" and "Syntax error" errors ...
```

- MPlayer may segfault when decoding and encoding video that uses the win32codecs:

```
...
Trying to force audio codec driver family acm...
Opening audio decoder: [acm] Win32/ACM decoders
sysi86(SI86DSCR): Invalid argument
Couldn't install fs segment, expect segfault


MPlayer interrupted by signal 11 in module: init_audio_codec
...
```

  This is because of a change to sysi86() in Solaris 10 and pre-Solaris Nevada b31 releases. This has been fixed in Solaris Nevada b32; however, Sun has yet to backport the fix to Solaris 10. The MPlayer Project has made Sun aware of the problem and a patch is currently in progress for Solaris 10. More information about this bug can be found at: http://bugs.opensolaris.org/bugdatabase/view_bug.do?bug_id=6308413.

- Due to bugs in Solaris 8, you may not be able to play DVD discs larger than 4 GB:

  - The sd(7D) driver on Solaris 8 x86 has a bug when accessing a disk block >4GB on a device using a logical blocksize != DEV_BSIZE (i.e. CD-ROM and DVD media). Due to a 32Bit int overflow, a disk address modulo 4GB is accessed (http://groups.yahoo.com/group/solarisonintel/message/22516). This problem does not exist in the SPARC version of Solaris 8.

  - A similar bug is present in the hsfs(7FS) file system code (AKA ISO9660), hsfs may not not support partitions/disks larger than 4GB, all data is accessed modulo 4GB (http://groups.yahoo.com/group/solarisonintel/message/22592). The hsfs problem can be fixed by installing patch 109764-04 (SPARC) / 109765-04 (x86).

9.3.2. HP-UX

Joe Page hosts a detailed HP-UX MPlayer HOWTO by Martin Gansser on his homepage. With these instructions the build should work out of the box. The following information is taken from this HOWTO.

You need GCC 3.4.0 or later and SDL 1.2.7 or later. HP cc will not produce a working program, prior GCC versions are buggy. For OpenGL functionality you need to install Mesa and the gl and gl2 video output drivers should work, speed may be very bad, depending on the CPU speed, though. A good replacement for the rather poor native HP-UX sound system is GNU esound.

Create the DVD device scan the SCSI bus with:

```
# ioscan -fn

Class          I          H/W   Path          Driver     S/W State     H/W Type        Des
...
```

```
ext_bus 1    8/16/5      c720  CLAIMED INTERFACE  Built-in SCSI
target  3    8/16/5.2    tgt   CLAIMED DEVICE
disk    4    8/16/5.2.0  sdisk CLAIMED DEVICE     PIONEER DVD-ROM DVD-305
                         /dev/dsk/c1t2d0 /dev/rdsk/c1t2d0
target  4    8/16/5.7    tgt   CLAIMED DEVICE
ctl     1    8/16/5.7.0  sctl  CLAIMED DEVICE     Initiator
                         /dev/rscsi/c1t7d0 /dev/rscsi/c1t7l0 /dev/scsi/c1t7l0
...
```

The screen output shows a Pioneer DVD-ROM at SCSI address 2. The card instance for hardware path 8/16 is 1.

Create a link from the raw device to the DVD device.

```
ln -s /dev/rdsk/c<SCSI bus instance>t<SCSI target ID>d<LUN> /dev/<device>
```

Example:

```
ln -s /dev/rdsk/c1t2d0 /dev/dvd
```

Below are solutions for some common problems:

- Crash at Start with the following error message:

  ```
  /usr/lib/dld.sl: Unresolved symbol: finite (code) from /usr/local/lib/gcc-lib/hppa2.0r
  ```

  This means that the function `.finite()`. is not available in the standard HP-UX math library. Instead there is `.isfinite()`.. Solution: Use the latest Mesa depot file.

- Crash at playback with the following error message:

  ```
  /usr/lib/dld.sl: Unresolved symbol: sem_init (code) from /usr/local/lib/libSDL-1.2.sl.
  ```

  Solution: Use the extralibdir option of configure `--with-extralibdir="/usr/lib -lrt"`

- MPlayer segfaults with a message like this:

  ```
  Pid 10166 received a SIGSEGV for stack growth failure.
  Possible causes: insufficient memory or swap space, or stack size exceeded maxssiz.
  Segmentation fault
  ```

  Solution: The HP-UX kernel has a default stack size of 8MB(?) per process.(11.0 and newer 10.20 patches let you increase `maxssiz` up to 350MB for 32-bit programs). You need to extend `maxssiz` and recompile the kernel (and reboot). You can use SAM to do this. (While at it, check out the `maxdsiz` parameter for the maximum amount of data a program can use. It depends on your applications, if the default of 64MB is enough or not.)

9.3.3. AIX

MPlayer builds successfully on AIX 5.1, 5.2, and 5.3, using GCC 3.3 or greater. Building MPlayer on AIX 4.3.3 and below is untested. It is highly recommended that you build MPlayer using GCC 3.4 or greater, or if you are building on POWER5, GCC 4.0 is required.

CPU detection is still a work in progress. The following architectures have been tested:

- 604e
- POWER3
- POWER4

The following architectures are untested, but should still work:

- POWER
- POWER2
- POWER5

Sound via the Ultimedia Services is not supported, as Ultimedia was dropped in AIX 5.1; therefore, the only option is to use the AIX Open Sound System (OSS) drivers from 4Front Technologies at http://www.opensound.com/aix.html. 4Front Technologies freely provides OSS drivers for AIX 5.1 for non-commercial use; however, there are currently no sound output drivers for AIX 5.2 or 5.3. This means **AIX 5.2 and 5.3 are not capable of MPlayer audio output, presently.**

Solutions to common problems:

- If you encounter this error message from `./configure`:

  ```
  $ ./configure
  ...
  Checking for iconv program ... no
  No working iconv program found, use
  --charset=US-ASCII to continue anyway.
  Messages in the GTK-2 interface will be broken then.
  ```

  This is because AIX uses non-standard character set names; therefore, converting MPlayer output to another character set is currently not supported. The solution is to use:

  ```
  $ ./configure --charset=noconv
  ```

9.3.4. QNX

You'll need to download and install SDL for QNX. Then run MPlayer with `-vo sdl:driver=photon` and `-ao sdl:nto` options, it should be fast.

The `-vo x11` output will be even slower than on Linux, since QNX has only X *emulation* which is very slow.

9.4. Windows

Yes, MPlayer runs on Windows under Cygwin and MinGW. It does not have an official GUI yet, but the command line version is completely functional. You should check out the MPlayer-cygwin mailing list for help and latest information. Official Windows binaries can be found on the download page. Installer packages and simple GUI frontends are available from external sources, we have collected then in the Windows section of our projects page.

If you wish to avoid using the command line, a simple trick is to put a shortcut on your desktop that contains something like the following in the execute section:

```
c:\path\to\mplayer.exe %1
```

This will make MPlayer play any movie that is dropped on the shortcut. Add `-fs` for fullscreen mode.

Best results are achieved with the native DirectX video output driver (`-vo directx`). Alternatives are OpenGL and SDL, but OpenGL performance varies greatly between systems and SDL is known to distort video or crash on some systems. If the image is distorted, try turning off hardware acceleration with `-vo directx:noaccel`. Download DirectX 7 header files to compile the DirectX video output driver. Furthermore you need to have DirectX 7 or later installed for the DirectX video output driver to work.

VIDIX now works under Windows as `-vo winvidix`, although it is still experimental and needs a bit of manual setup. Download dhahelper.sys or dhahelper.sys (with MTRR support) and copy it to the `vidix/dhahelperwin` directory in your MPlayer source tree. Open a console and type

```
make install-dhahelperwin
```

as Administrator. After that you will have to reboot.

For best results MPlayer should use a colorspace that your video card supports in hardware. Unfortunately many Windows graphics drivers wrongly report some colorspaces as supported in hardware. To find out which, try

```
mplayer -benchmark -nosound -frames 100 -vf format=colorspace movie
```

where *colorspace* can be any colorspace printed by the `-vf format=fmt=help` option. If you find a colorspace your card handles particularly bad `-vf noformat=colorspace` will keep it from being used. Add this to your config file to permanently keep it from being used.

There are special codec packages for Windows available on our download page to allow playing formats for which there is no native support yet. Put the codecs somewhere in your path or pass `--codecsdir=c:/path/to/your/codecs` (alternatively `--codecsdir=/path/to/your/codecs` only on Cygwin) to `configure`. We have had some reports that Real DLLs need to be writable by the user running MPlayer, but only on some systems (NT4). Try making them writable if you have problems.

You can play VCDs by playing the `.DAT` or `.MPG` files that Windows exposes on VCDs. It works like this (adjust for the drive letter of your CD-ROM):

```
mplayer d:/mpegav/avseq01.dat
```

Alternatively, you can play a VCD track directly by using:

```
mplayer vcd://<track> -cdrom-device d:
```

DVDs also work, adjust `-dvd-device` for the drive letter of your DVD-ROM:

```
mplayer dvd://<title> -dvd-device d:
```

The Cygwin/MinGW console is rather slow. Redirecting output or using the `-quiet` option has been reported to improve performance on some systems. Direct rendering (`-dr`) may also help. If playback is jerky, try `-autosync 100`. If some of these options help you, you may want to put them in your config file.

Note

If you have a Pentium 4 and are experiencing a crash using the RealPlayer codecs, you may need to disable hyperthreading support.

9.4.1. Cygwin

You need to run Cygwin 1.5.0 or later in order to compile MPlayer.

DirectX header files need to be extracted to `/usr/include/` or `/usr/local/include/`.

Instructions and files for making SDL run under Cygwin can be found on the libsdl site.

9.4.2. MinGW

You need MinGW 3.1.0 or later and MSYS 1.0.9 or later. Tell the MSYS postinstall that MinGW is installed.

Extract DirectX header files to `/mingw/include/`.

MOV compressed header support requires zlib, which MinGW does not provide by default. Configure it with `--prefix=/mingw` and install it before compiling MPlayer.

Complete instructions for building MPlayer and necessary libraries can be found in the MPlayer MinGW HOWTO.

9.5. Mac OS

MPlayer does not work on Mac OS versions before 10, but should compile out-of-the-box on Mac OS X 10.2 and up. The preferred compiler is the Apple version of GCC 3.x or later. You can get the basic compilation environment by installing Apple's Xcode. If you have Mac OS X 10.3.9 or later and QuickTime 7 you can use the `corevideo` video output driver.

Unfortunately, this basic environment will not allow you to take advantage of all the nice features of MPlayer. For instance, in order to have OSD support compiled in, you will need to have `fontconfig` and `freetype` libraries installed on your machine. Contrary to other Unixes such as most Linux and BSD variants, OS X does not have a package system that comes with the system.

There are at least two to choose from: Fink and MacPorts. Both of them provide about the same service (i.e. a lot of packages to choose from, dependency resolution, the ability to simply add/update/remove packages, etc...). Fink offers both precompiled binary packages or building everything from source, whereas MacPorts only offers building from source. The author of this guide chose MacPorts for the simple fact that its basic setup was more lightweight. Later examples will be based on MacPorts.

For instance, to compile MPlayer with OSD support:

```
sudo port install pkg-config
```

This will install pkg-config, which is a system for managing library compile/link flags. MPlayer's `configure` script uses it to properly detect libraries. Then you can install fontconfig in a similar way:

```
sudo port install fontconfig
```

Then you can proceed with launching MPlayer's `configure` script (note the `PKG_CONFIG_PATH` and `PATH` environment variables so that `configure` finds the libraries installed with MacPorts):

```
PKG_CONFIG_PATH=/opt/local/lib/pkgconfig/ PATH=$PATH:/opt/local/bin/ ./configure
```

9.5.1. MPlayer OS X GUI

You can get a native GUI for MPlayer together with precompiled MPlayer binaries for Mac OS X from the MPlayerOSX project, but be warned: that project is not active anymore.

Fortunately, MPlayerOSX has been taken over by a member of the MPlayer team. Preview releases are available from our download page and an official release should arrive soon.

In order to build MPlayerOSX from source yourself, you need the `mplayerosx`, the `main` and a copy of the `main` SVN module named `main_noaltivec`. `mplayerosx` is the GUI frontend, `main` is MPlayer and `main_noaltivec` is MPlayer built without AltiVec support.

To check out SVN modules use:

```
svn checkout svn://svn.mplayerhq.hu/mplayerosx/trunk/ mplayerosx
svn checkout svn://svn.mplayerhq.hu/mplayer/trunk/ main
```

In order to build MPlayerOSX you will need to set up something like this:

```
MPlayer_source_directory
   |
   |--->main           (MPlayer Subversion source)
   |
   |--->main_noaltivec (MPlayer Subversion source configured with --disable-altivec)
   |
   \--->mplayerosx     (MPlayer OS X Subversion source)
```

You first need to build main and main_noaltivec.

To begin with, in order to ensure maximum backwards compatibility, set an environment variable:

```
export MACOSX_DEPLOYMENT_TARGET=10.3
```

Then, configure:

If you configure for a G4 or later CPU with AltiVec support, do as follows:

```
./configure --disable-gl --disable-x11
```

If you configure for a G3-powered machine without AltiVec, use:

```
./configure --disable-gl --disable-x11 --disable-altivec
```

You may need to edit `config.mak` and change `-mcpu` and `-mtune` from `74XX` to `G3`.

Continue with

```
make
```

then go to the mplayerosx directory and type

```
make dist
```

This will create a compressed `.dmg` archive with the ready to use binary.

You can also use the Xcode 2.1 project; the old project for Xcode 1.x does not work anymore.

Chapter 10. Basic usage of MEncoder

For the complete list of available MEncoder options and examples, please see the man page. For a series of hands-on examples and detailed guides on using several encoding parameters, read the encoding-tips that were collected from several mailing list threads on MPlayer-users. Search the archives here and especially for older things also here for a wealth of discussions about all aspects of and problems related to encoding with MEncoder.

10.1. Selecting codecs and container formats

Audio and video codecs for encoding are selected with the `-oac` and `-ovc` options, respectively. Type for instance:

```
mencoder -ovc help
```

to list all video codecs supported by the version of MEncoder on your machine. The following choices are available:

Audio Codecs:

Audio codec name	Description
mp3lame	encode to VBR, ABR or CBR MP3 with LAME
lavc	use one of `libavcodec`'s audio codecs
faac	FAAC AAC audio encoder
toolame	MPEG Audio Layer 2 encoder
twolame	MPEG Audio Layer 2 encoder based on tooLAME
pcm	uncompressed PCM audio
copy	do not reencode, just copy compressed frames

Video codecs:

Video codec name	Description
lavc	use one of `libavcodec`'s video codecs
xvid	Xvid, MPEG-4 Advanced Simple Profile (ASP) codec
x264	x264, MPEG-4 Advanced Video Coding (AVC), AKA H.264 codec
nuv	nuppel video, used by some realtime applications
raw	uncompressed video frames
copy	do not reencode, just copy compressed frames
frameno	used for 3-pass encoding (not recommended)

Output container formats are selected with the `-of` option. Type:

```
mencoder -of help
```

to list all containers supported by the version of MEncoder on your machine. The following choices are available:

Container formats:

Container format name	Description
lavf	one of the containers supported by `libavformat`
avi	Audio-Video Interleaved
mpeg	MPEG-1 and MPEG-2 PS
rawvideo	raw video stream (no muxing - one video stream only)
rawaudio	raw audio stream (no muxing - one audio stream only)

The AVI container is the native container format for MEncoder, which means that it's the one that is best handled, and the one for which MEncoder was designed. As noted above, other container formats are usable, but you may experience problems when using them.

`libavformat` containers:

If you selected `libavformat` to do the muxing of the output file (by using the `-of lavf`), the appropriate container format will be determined by the file extension of the output file. You may force a particular container format with `libavformat`'s `format` option.

`libavformat` container name	Description
mpg	MPEG-1 and MPEG-2 PS
asf	Advanced Streaming Format
avi	Audio-Video Interleaved
wav	Waveform Audio
swf	Macromedia Flash
flv	Macromedia Flash video

rm	RealMedia
au	SUN AU
nut	NUT open container (experimental and not yet spec-compliant)
mov	QuickTime
mp4	MPEG-4 format
dv	Sony Digital Video container
mkv	Matroska open audio/video container

As you can see, `libavformat` allows MEncoder to mux into a considerable variety of containers. Unfortunately, as MEncoder was not designed from the beginning to support container formats other than AVI, your should really be paranoid about the resulting file. Please check to be sure that the audio/video synchronization is OK and that the file can be played correctly by players other than MPlayer.

Example 10.1. encode to Macromedia Flash format

Creating a Macromedia Flash video suitable for playback in a web browser with the Macromedia Flash plugin:

```
mencoder input.avi -o output.flv -of lavf \
    -oac mp3lame -lameopts abr:br=56 -srate 22050 -ovc lavc \
    -lavcopts vcodec=flv:vbitrate=500:mbd=2:mv0:trell:v4mv:cbp:last_pred=3
```

10.2. Selecting input file or device

MEncoder can encode from files or directly from a DVD or VCD disc. Simply include the filename on the command line to encode from a file, or `dvd://`*`titlenumber`* or `vcd://`*`tracknumber`* to encode from a DVD title or VCD track. If you have already copied a DVD to your hard drive (you can use a tool such as dvdbackup, available on most systems), and wish to encode from the copy, you should still use the `dvd://` syntax, along with `-dvd-device` followed by the path to the copied DVD root. The `-dvd-device` and `-cdrom-device` options can also be used to override the paths to the device nodes for reading directly from disc, if the defaults of `/dev/dvd` and `/dev/cdrom` do not work on your system.

When encoding from DVD, it is often desirable to select a chapter or range of chapters to encode. You can use the `-chapter` option for this purpose. For example, `-chapter` *`1-4`* will only encode chapters 1 through 4 from the DVD. This is especially useful if you will be making a 1400 MB encode targeted for two CDs, since you can ensure the split occurs exactly at a chapter boundary rather than in the middle of a scene.

If you have a supported TV capture card, you can also encode from the TV-in device. Use `tv://`*`channelnumber`* as the filename, and `-tv` to configure various capture settings. DVB input works similarly.

10.3. Encoding two pass MPEG-4 ("DivX")

The name comes from the fact that this method encodes the file *twice*. The first encoding (dubbed pass) creates some temporary files (`*.log`) with a size of few megabytes, do not delete them yet (you can delete the AVI or rather just not create any video by redirecting it into `/dev/null` or on Windows into `NUL`). In the second pass, the two pass output file is created, using the bitrate data from the temporary files. The resulting file will have much better image quality. If this is the first time you heard about this, you should consult some guides available on the net.

Example 10.2. copy audio track

Two pass encode of the second track a DVD to an MPEG-4 ("DivX") AVI while copying the audio track.

```
mencoder dvd://2 -ovc lavc -lavcopts vcodec=mpeg4:vpass=1 -oac copy -o /dev/null
mencoder dvd://2 -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell:vpass=2 \
    -oac copy -o output.avi
```

Example 10.3. encode audio track

Two pass encode of a DVD to an MPEG-4 ("DivX") AVI while encoding the audio track to MP3. Be careful using this method as it may lead to audio/video desync in some cases.

```
mencoder dvd://2 -ovc lavc -lavcopts vcodec=mpeg4:vpass=1 \
    -oac mp3lame -lameopts vbr=3 -o /dev/null
mencoder dvd://2 -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell:vpass=2 \
    -oac mp3lame -lameopts vbr=3 -o output.avi
```

10.4. Encoding to Sony PSP video format

MEncoder supports encoding to Sony PSP's video format, but, depending on the revision of the PSP software, the constraints may differ. You should be safe if you respect the following constraints:

- **Bitrate**: it should not exceed 1500kbps, however, past versions supported pretty much any bitrate as long as the header claimed it was not too high.
- **Dimensions**: the width and height of the PSP video should be multiples of 16, and the product width * height must be <= 64000. Under some circumstances, it may be possible for the PSP to play higher resolutions.
- **Audio**: its samplerate should be 24kHz for MPEG-4 videos, and 48kHz for H.264.

Example 10.4. encode for PSP

```
mencoder -ofps 30000/1001 -af lavcresample=24000 -vf harddup -of lavf \
    -oac lavc -ovc lavc -lavcopts aglobal=1:vglobal=1:vcodec=mpeg4:acodec=aac \
    -lavfopts format=psp \
    input.video -o output.psp
```

Note that you can set the title of the video with `-info name=MovieTitle`.

10.5. Encoding to MPEG format

MEncoder can create MPEG (MPEG-PS) format output files. Usually, when you are using MPEG-1 or MPEG-2 video, it is because you are encoding for a constrained format such as SVCD, VCD, or DVD. The specific requirements for these formats are explained in the VCD and DVD creation guide section.

To change MEncoder's output file format, use the `-of mpeg` option.

Example:

```
mencoder input.avi -of mpeg -ovc lavc -lavcopts vcodec=mpeg1video \
    -oac copy other_options -o output.mpg
```

Creating an MPEG-1 file suitable to be played on systems with minimal multimedia support, such as default Windows installs:

```
mencoder input.avi -of mpeg -mpegopts format=mpeg1:tsaf:muxrate=2000 \
    -o output.mpg -oac lavc -lavcopts acodec=mp2:abitrate=224 -ovc lavc \
    -lavcopts vcodec=mpeg1video:vbitrate=1152:keyint=15:mbd=2:aspect=4/3
```

Same, but using `libavformat` MPEG muxer:

```
mencoder input.avi -o VCD.mpg -ofps 25 -vf scale=352:288,harddup -of lavf \
    -lavfopts format=mpg -oac lavc -lavcopts acodec=mp2:abitrate=224 -ovc lavc \
    -lavcopts vcodec=mpeg1video:vrc_buf_size=327:keyint=15:vrc_maxrate=1152:vbitrate=1152:
```

Hint:

If for some reason the video quality of the second pass did not satisfy you, you may re-run your video encode with a different target bitrate, provided that you saved the statistics file of the previous pass. This is possible because the statistics file's primary goal is to record the complexity of each frame, which doesn't depend heavily on bitrate. You should note, though, that you'll get the best results if all passes are run with target bitrates that do not differ very much.

10.6. Rescaling movies

Often the need to resize movie images emerges. The reasons can be many: decreasing file size, network bandwidth, etc. Most people even do rescaling when converting DVDs or SVCDs to DivX AVI. If you wish to rescale, read the Preserving aspect ratio section.

The scaling process is handled by the `scale` video filter: `-vf scale=`*`width`*`:`*`height`*. Its quality can be set with the `-sws` option. If it is not specified, MEncoder will use 2: bicubic.

Usage:

```
mencoder input.mpg -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell \
    -vf scale=640:480 -o output.avi
```

10.7. Stream copying

MEncoder can handle input streams in two ways: **encode** or **copy** them. This section is about **copying**.

- **Video stream** (option `-ovc copy`): nice stuff can be done :) Like, putting (not converting!) FLI or VIVO or MPEG-1 video into an AVI file! Of course only MPlayer can play such files :) And it probably has no real life value at all. Rationally: video stream copying can be useful for example when only the audio stream has to be encoded (like, uncompressed PCM to MP3).

- **Audio stream** (option `-oac copy`): straightforward. It is possible to take an external audio file (MP3, WAV) and mux it into the output stream. Use the `-audiofile` *`filename`* option for this.

Using `-oac copy` to copy from one container format to another may require the use of `-fafmttag` to keep the audio format tag of the original file. For example, if you are converting an NSV file with AAC audio to an AVI container, the audio format tag will be incorrect and it will have to be changed. For a list of audio format tags, check `codecs.conf`.

Example:

```
mencoder input.nsv -oac copy -fafmttag 0x706D \
    -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell -o output.avi
```

10.8. Encoding from multiple input image files (JPEG, PNG, TGA, etc.)

MEncoder is capable of creating movies from one or more JPEG, PNG, TGA, or other image files. With simple framecopy it can create MJPEG (Motion JPEG), MPNG (Motion PNG) or MTGA (Motion TGA) files.

Explanation of the process:

1. MEncoder *decodes* the input image(s) with `libjpeg` (when decoding PNGs, it will use `libpng`).

2. MEncoder then feeds the decoded image to the chosen video compressor (DivX4, Xvid, FFmpeg msmpeg4, etc.).

Examples. The explanation of the `-mf` option is in the man page.

Creating an MPEG-4 file from all the JPEG files in the current directory:

```
mencoder mf://*.jpg -mf w=800:h=600:fps=25:type=jpg -ovc lavc \
    -lavcopts vcodec=mpeg4:mbd=2:trell -oac copy -o output.avi
```

Creating an MPEG-4 file from some JPEG files in the current directory:

```
mencoder mf://frame001.jpg,frame002.jpg -mf w=800:h=600:fps=25:type=jpg \
    -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell -oac copy -o output.avi
```

Creating an MPEG-4 file from explicit list of JPEG files (list.txt in current directory contains the list of files to use as source, one per line):

```
mencoder mf://@list.txt -mf w=800:h=600:fps=25:type=jpg \
    -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell -oac copy -o output.avi
```

You can mix different types of images, regardless of the method you use — individual filenames, wildcard or file with list — provided of course they have the same dimensions. So you can e.g. take title frame from PNG file, and then put a slideshow of your JPEG photos.

Creating a Motion JPEG (MJPEG) file from all the JPEG files in the current directory:

```
mencoder mf://*.jpg -mf w=800:h=600:fps=25:type=jpg -ovc copy -oac copy -o output.avi
```

Creating an uncompressed file from all the PNG files in the current directory:

```
mencoder mf://*.png -mf w=800:h=600:fps=25:type=png -ovc raw -oac copy -o output.avi
```

Note

Width must be integer multiple of 4, it is a limitation of the RAW RGB AVI format.

Creating a Motion PNG (MPNG) file from all the PNG files in the current directory:

```
mencoder mf://*.png -mf w=800:h=600:fps=25:type=png -ovc copy -oac copy -o output.avi
```

Creating a Motion TGA (MTGA) file from all the TGA files in the current directory:

```
mencoder mf://*.tga -mf w=800:h=600:fps=25:type=tga -ovc copy -oac copy -o output.avi
```

10.9. Extracting DVD subtitles to VOBsub file

MEncoder is capable of extracting subtitles from a DVD into VOBsub formatted files. They consist of a pair of files ending in `.idx` and `.sub` and are usually packaged in a single `.rar` archive. MPlayer can play these with the `-vobsub` and `-vobsubid` options.

You specify the basename (i.e without the `.idx` or `.sub` extension) of the output files with `-vobsubout` and the index for this subtitle in the resulting files with `-vobsuboutindex`.

If the input is not from a DVD you should use `-ifo` to indicate the `.ifo` file needed to construct the resulting `.idx` file.

If the input is not from a DVD and you do not have the `.ifo` file you will need to use the `-vobsubid` option to let it know what language id to put in the `.idx` file.

Each run will append the running subtitle if the `.idx` and `.sub` files already exist. So you should remove any before starting.

Example 10.5. Copying two subtitles from a DVD while doing two pass encoding

```
rm subtitles.idx subtitles.sub
mencoder dvd://1 -oac copy -ovc lavc -lavcopts vcodec=mpeg4:vpass=1 \
    -vobsubout subtitles -vobsuboutindex 0 -sid 2
mencoder dvd://1 -oac copy -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell:vpass=2 \
    -vobsubout subtitles -vobsuboutindex 1 -sid 5
```

Example 10.6. Copying a French subtitle from an MPEG file

```
rm subtitles.idx subtitles.sub
mencoder movie.mpg -ifo movie.ifo -vobsubout subtitles -vobsuboutindex 0 \
    -vobsuboutid fr -sid 1 -nosound -ovc copy
```

10.10. Preserving aspect ratio

DVDs and SVCDs (i.e. MPEG-1/2) files contain an aspect ratio value, which describes how the player should scale the video stream, so humans will not have egg heads (ex.: 480x480 + 4:3 = 640x480). However when encoding to AVI (DivX) files, you have to be aware that AVI headers do not store this value. Rescaling the movie is disgusting and time consuming, there has to be a better way!

There is

MPEG-4 has a unique feature: the video stream can contain its needed aspect ratio. Yes, just like MPEG-1/2 (DVD, SVCD) and H.263 files. Regretfully, there are few video players apart from MPlayer that support this MPEG-4 attribute.

This feature can be used only with `libavcodec`'s `mpeg4` codec. Keep in mind: although MPlayer will correctly play the created file, other players may use the wrong aspect ratio.

You seriously should crop the black bands over and below the movie image. See the man page for the usage of the `cropdetect` and `crop` filters.

Usage

```
mencoder sample-svcd.mpg -vf crop=714:548:0:14 -oac copy -ovc lavc \
    -lavcopts vcodec=mpeg4:mbd=2:trell:autoaspect -o output.avi
```

Chapter 11. Encoding with MEncoder

11.1. Making a high quality MPEG-4 ("DivX") rip of a DVD movie

One frequently asked question is "How do I make the highest quality rip for a given size?". Another question is "How do I make the highest quality DVD rip possible? I do not care about file size, I just want the best quality."

The latter question is perhaps at least somewhat wrongly posed. After all, if you do not care about file size, why not simply copy the entire MPEG-2 video stream from the the DVD? Sure, your AVI will end up being 5GB, give or take, but if you want the best quality and do not care about size, this is certainly your best option.

In fact, the reason you want to transcode a DVD into MPEG-4 is specifically because you **do** care about file size.

It is difficult to offer a cookbook recipe on how to create a very high quality DVD rip. There are several factors to consider, and you should understand these details or else you are likely to end up disappointed with your results. Below we will investigate some of these issues, and then have a look at an example. We assume you are using `libavcodec` to encode the video, although the theory applies to other codecs as well.

If this seems to be too much for you, you should probably use one of the many fine frontends that are listed in the MEncoder section of our related projects page. That way, you should be able to achieve high quality rips without too much thinking, because most of those tools are designed to take clever decisions for you.

11.1.1. Preparing to encode: Identifying source material and framerate

Before you even think about encoding a movie, you need to take several preliminary steps.

The first and most important step before you encode should be determining what type of content you are dealing with. If your source material comes from DVD or broadcast/cable/satellite TV, it will be stored in one of two formats: NTSC for North America and Japan, PAL for Europe, etc. It is important to realize, however, that this is just the formatting for presentation on a television, and often does **not** correspond to the original format of the movie. Experience shows that NTSC material is a lot more difficult to encode, because there more elements to identify in the source. In order to produce a suitable encode, you need to know the original format. Failure to take this into account will result in various flaws in your encode, including ugly combing (interlacing) artifacts and duplicated or even lost frames. Besides being ugly, the artifacts also harm coding efficiency: You will get worse quality per unit bitrate.

11.1.1.1. Identifying source framerate

Here is a list of common types of source material, where you are likely to find them, and their properties:

- **Standard Film**: Produced for theatrical display at 24fps.
- **PAL video**: Recorded with a PAL video camera at 50 fields per second. A field consists of just the odd- or even-numbered lines of a frame. Television was designed to refresh these in alternation as a cheap form of analog compression. The human eye supposedly compensates for this, but once you understand interlacing you will learn to see it on TV too and never enjoy TV again. Two fields do **not** make a complete frame, because they are captured 1/50 of a second apart in time, and thus they do not line up unless there is no motion.
- **NTSC Video**: Recorded with an NTSC video camera at 60000/1001 fields per second, or 60 fields per second in the pre-color era. Otherwise similar to PAL.
- **Animation**: Usually drawn at 24fps, but also comes in mixed-framerate varieties.
- **Computer Graphics (CG)**: Can be any framerate, but some are more common than others; 24 and 30 frames per

second are typical for NTSC, and 25fps is typical for PAL.

- **Old Film**: Various lower framerates.

11.1.1.2. Identifying source material

Movies consisting of frames are referred to as progressive, while those consisting of independent fields are called either interlaced or video - though this latter term is ambiguous.

To further complicate matters, some movies will be a mix of several of the above.

The most important distinction to make between all of these formats is that some are frame-based, while others are field-based. **Whenever** a movie is prepared for display on television (including DVD), it is converted to a field-based format. The various methods by which this can be done are collectively referred to as "telecine", of which the infamous NTSC "3:2 pulldown" is one variety. Unless the original material was also field-based (and the same fieldrate), you are getting the movie in a format other than the original.

There are several common types of pulldown:

- **PAL 2:2 pulldown**: The nicest of them all. Each frame is shown for the duration of two fields, by extracting the even and odd lines and showing them in alternation. If the original material is 24fps, this process speeds up the movie by 4%.

- **PAL 2:2:2:2:2:2:2:2:2:2:2:3 pulldown**: Every 12th frame is shown for the duration of three fields, instead of just two. This avoids the 4% speedup issue, but makes the process much more difficult to reverse. It is usually seen in musical productions where adjusting the speed by 4% would seriously damage the musical score.

- **NTSC 3:2 telecine**: Frames are shown alternately for the duration of 3 fields or 2 fields. This gives a fieldrate 2.5 times the original framerate. The result is also slowed down very slightly from 60 fields per second to 60000/1001 fields per second to maintain NTSC fieldrate.

- **NTSC 2:2 pulldown**: Used for showing 30fps material on NTSC. Nice, just like 2:2 PAL pulldown.

There are also methods for converting between NTSC and PAL video, but such topics are beyond the scope of this guide. If you encounter such a movie and want to encode it, your best bet is to find a copy in the original format. Conversion between these two formats is highly destructive and cannot be reversed cleanly, so your encode will greatly suffer if it is made from a converted source.

When video is stored on DVD, consecutive pairs of fields are grouped as a frame, even though they are not intended to be shown at the same moment in time. The MPEG-2 standard used on DVD and digital TV provides a way both to encode the original progressive frames and to store the number of fields for which a frame should be shown in the header of that frame. If this method has been used, the movie will often be described as "soft-telecined", since the process only directs the DVD player to apply pulldown to the movie rather than altering the movie itself. This case is highly preferable since it can easily be reversed (actually ignored) by the encoder, and since it preserves maximal quality. However, many DVD and broadcast production studios do not use proper encoding techniques but instead produce movies with "hard telecine", where fields are actually duplicated in the encoded MPEG-2.

The procedures for dealing with these cases will be covered later in this guide. For now, we leave you with some guides to identifying which type of material you are dealing with:

NTSC regions:

- If MPlayer prints that the framerate has changed to 24000/1001 when watching your movie, and never changes back, it is almost certainly progressive content that has been "soft telecined".

- If MPlayer shows the framerate switching back and forth between 24000/1001 and 30000/1001, and you see "combing" at times, then there are several possibilities. The 24000/1001 fps segments are almost certainly progressive content, "soft telecined", but the 30000/1001 fps parts could be either hard-telecined 24000/1001 fps content or 60000/1001 fields per second NTSC video. Use the same guidelines as the following two cases to determine which.

- If MPlayer never shows the framerate changing, and every single frame with motion appears combed, your movie is NTSC video at 60000/1001 fields per second.

- If MPlayer never shows the framerate changing, and two frames out of every five appear combed, your movie is "hard telecined" 24000/1001fps content.

PAL regions:

- If you never see any combing, your movie is 2:2 pulldown.
- If you see combing alternating in and out every half second, then your movie is 2:2:2:2:2:2:2:2:2:2:2:3 pulldown.
- If you always see combing during motion, then your movie is PAL video at 50 fields per second.

Hint:

MPlayer can slow down movie playback with the -speed option or play it frame-by-frame. Try using `-speed` 0.2 to watch the movie very slowly or press the "." key repeatedly to play one frame at a time and identify the pattern, if you cannot see it at full speed.

11.1.2. Constant quantizer vs. multipass

It is possible to encode your movie at a wide range of qualities. With modern video encoders and a bit of pre-codec compression (downscaling and denoising), it is possible to achieve very good quality at 700 MB, for a 90-110 minute widescreen movie. Furthermore, all but the longest movies can be encoded with near-perfect quality at 1400 MB.

There are three approaches to encoding the video: constant bitrate (CBR), constant quantizer, and multipass (ABR, or average bitrate).

The complexity of the frames of a movie, and thus the number of bits required to compress them, can vary greatly from one scene to another. Modern video encoders can adjust to these needs as they go and vary the bitrate. In simple modes such as CBR, however, the encoders do not know the bitrate needs of future scenes and so cannot exceed the requested average bitrate for long stretches of time. More advanced modes, such as multipass encode, can take into account the statistics from previous passes; this fixes the problem mentioned above.

Note:

Most codecs which support ABR encode only support two pass encode while some others such as `x264`, `Xvid` and `libavcodec` support multipass, which slightly improves quality at each pass, yet this improvement is no longer measurable nor noticeable after the 4th or so pass. Therefore, in this section, two pass and multipass will be used interchangeably.

In each of these modes, the video codec (such as `libavcodec`) breaks the video frame into 16x16 pixel macroblocks and then applies a quantizer to each macroblock. The lower the quantizer, the better the quality and higher the bitrate. The method the movie encoder uses to determine which quantizer to use for a given macroblock varies and is highly tunable. (This is an extreme over-simplification of the actual process, but the basic concept is useful to understand.)

When you specify a constant bitrate, the video codec will encode the video, discarding detail as much as necessary and as little as possible in order to remain lower than the given bitrate. If you truly do not care about file size, you could as well use CBR and specify a bitrate of infinity. (In practice, this means a value high enough so that it poses no limit, like 10000Kbit.) With no real restriction on bitrate, the result is that the codec will use the lowest possible quantizer for each macroblock (as specified by `vqmin` for `libavcodec`, which is 2 by default). As soon as you specify a low enough bitrate that the codec is forced to use a higher quantizer, then you are almost certainly ruining the quality of your video. In order to avoid that, you should probably downscale your video, according to the method described later on in this guide. In general, you should avoid CBR altogether if you care about quality.

With constant quantizer, the codec uses the same quantizer, as specified by the `vqscale` option (for `libavcodec`), on every macroblock. If you want the highest quality rip possible, again ignoring bitrate, you can use `vqscale=2`. This will yield the same bitrate and PSNR (peak signal-to-noise ratio) as CBR with `vbitrate`=infinity and the default `vqmin` of 2.

The problem with constant quantizing is that it uses the given quantizer whether the macroblock needs it or not. That is, it might be possible to use a higher quantizer on a macroblock without sacrificing visual quality. Why waste the bits on an unnecessarily low quantizer? Your CPU has as many cycles as there is time, but there is only so many bits on your hard disk.

With a two pass encode, the first pass will rip the movie as though it were CBR, but it will keep a log of properties for each frame. This data is then used during the second pass in order to make intelligent decisions about which quantizers to use. During fast action or high detail scenes, higher quantizers will likely be used, and during slow moving or low detail scenes, lower quantizers will be used. Normally, the amount of motion is much more important than the amount of detail.

If you use `vqscale=2`, then you are wasting bits. If you use `vqscale=3`, then you are not getting the highest quality rip. Suppose you rip a DVD at `vqscale=3`, and the result is 1800Kbit. If you do a two pass encode with `vbitrate=1800`, the resulting video will have **higher quality** for the **same bitrate**.

Since you are now convinced that two pass is the way to go, the real question now is what bitrate to use? The answer is that there is no single answer. Ideally you want to choose a bitrate that yields the best balance between quality and file size. This is going to vary depending on the source video.

If size does not matter, a good starting point for a very high quality rip is about 2000Kbit plus or minus 200Kbit. For fast action or high detail source video, or if you just have a very critical eye, you might decide on 2400 or 2600. For some DVDs, you might not notice a difference at 1400Kbit. It is a good idea to experiment with scenes at different bitrates to get a feel.

If you aim at a certain size, you will have to somehow calculate the bitrate. But before that, you need to know how much space you should reserve for the audio track(s), so you should rip those first. You can compute the bitrate with the following equation: `bitrate = (target_size_in_Mbytes - sound_size_in_Mbytes) * 1024 * 1024 / length_in_secs * 8 / 1000` For instance, to squeeze a two-hour movie onto a 702MB CD, with 60MB of audio track, the video bitrate will have to be: `(702 - 60) * 1024 * 1024 / (120*60) * 8 / 1000 = 740kbps`

11.1.3. Constraints for efficient encoding

Due to the nature of MPEG-type compression, there are various constraints you should follow for maximal quality. MPEG splits the video up into 16x16 squares called macroblocks, each composed of 4 8x8 blocks of luma (intensity) information and two half-resolution 8x8 chroma (color) blocks (one for red-cyan axis and the other for the blue-yellow axis). Even if your movie width and height are not multiples of 16, the encoder will use enough 16x16 macroblocks to cover the whole picture area, and the extra space will go to waste. So in the interests of maximizing quality at a fixed file size, it is a bad idea to use dimensions that are not multiples of 16.

Most DVDs also have some degree of black borders at the edges. Leaving these in place will hurt quality **a lot** in several ways.

1. MPEG-type compression is highly dependent on frequency domain transformations, in particular the Discrete Cosine Transform (DCT), which is similar to the Fourier transform. This sort of encoding is efficient for representing patterns and smooth transitions, but it has a hard time with sharp edges. In order to encode them it must use many more bits, or else an artifact known as ringing will appear.

   The frequency transform (DCT) takes place separately on each macroblock (actually each block), so this problem only applies when the sharp edge is inside a block. If your black borders begin exactly at multiple-of-16 pixel boundaries, this is not a problem. However, the black borders on DVDs rarely come nicely aligned, so in practice you will always need to crop to avoid this penalty.

In addition to frequency domain transforms, MPEG-type compression uses motion vectors to represent the change from one frame to the next. Motion vectors naturally work much less efficiently for new content coming in from the edges of the picture, because it is not present in the previous frame. As long as the picture extends all the way to the edge of the encoded region, motion vectors have no problem with content moving out the edges of the picture. However, in the presence of black borders, there can be trouble:

2. For each macroblock, MPEG-type compression stores a vector identifying which part of the previous frame should be copied into this macroblock as a base for predicting the next frame. Only the remaining differences need to be encoded. If a macroblock spans the edge of the picture and contains part of the black border, then motion vectors from other parts of the picture will overwrite the black border. This means that lots of bits must be spent either re-blackening the border that was overwritten, or (more likely) a motion vector will not be used at all and all the changes in this macroblock will have to be coded explicitly. Either way, encoding efficiency is greatly reduced.

   Again, this problem only applies if black borders do not line up on multiple-of-16 boundaries.

3. Finally, suppose we have a macroblock in the interior of the picture, and an object is moving into this block from near the edge of the image. MPEG-type coding cannot say "copy the part that is inside the picture but not the black border." So the black border will get copied inside too, and lots of bits will have to be spent encoding the part of the

picture that is supposed to be there.

If the picture runs all the way to the edge of the encoded area, MPEG has special optimizations to repeatedly copy the pixels at the edge of the picture when a motion vector comes from outside the encoded area. This feature becomes useless when the movie has black borders. Unlike problems 1 and 2, aligning the borders at multiples of 16 does not help here.

4. Despite the borders being entirely black and never changing, there is at least a minimal amount of overhead involved in having more macroblocks.

For all of these reasons, it is recommended to fully crop black borders. Further, if there is an area of noise/distortion at the edge of the picture, cropping this will improve encoding efficiency as well. Videophile purists who want to preserve the original as close as possible may object to this cropping, but unless you plan to encode at constant quantizer, the quality you gain from cropping will considerably exceed the amount of information lost at the edges.

11.1.4. Cropping and Scaling

Recall from the previous section that the final picture size you encode should be a multiple of 16 (in both width and height). This can be achieved by cropping, scaling, or a combination of both.

When cropping, there are a few guidelines that must be followed to avoid damaging your movie. The normal YUV format, 4:2:0, stores chroma (color) information subsampled, i.e. chroma is only sampled half as often in each direction as luma (intensity) information. Observe this diagram, where L indicates luma sampling points and C chroma.

```
L   L   L   L   L   L   L   L
  C       C       C       C
L   L   L   L   L   L   L   L
L   L   L   L   L   L   L   L
  C       C       C       C
L   L   L   L   L   L   L   L
```

As you can see, rows and columns of the image naturally come in pairs. Thus your crop offsets and dimensions *must* be even numbers. If they are not, the chroma will no longer line up correctly with the luma. In theory, it is possible to crop with odd offsets, but it requires resampling the chroma which is potentially a lossy operation and not supported by the crop filter.

Further, interlaced video is sampled as follows:

```
             Top field                          Bottom field
L   L   L   L   L   L   L   L
  C       C       C       C
                                L   L   L   L   L   L   L   L
L   L   L   L   L   L   L   L
                                  C       C       C       C
                                L   L   L   L   L   L   L   L
L   L   L   L   L   L   L   L
  C       C       C       C
                                L   L   L   L   L   L   L   L
L   L   L   L   L   L   L   L
                                  C       C       C       C
                                L   L   L   L   L   L   L   L
```

As you can see, the pattern does not repeat until after 4 lines. So for interlaced video, your y-offset and height for cropping must be multiples of 4.

Native DVD resolution is 720x480 for NTSC, and 720x576 for PAL, but there is an aspect flag that specifies whether it is full-screen (4:3) or wide-screen (16:9). Many (if not most) widescreen DVDs are not strictly 16:9, and will be either 1.85:1 or 2.35:1 (cinescope). This means that there will be black bands in the video that will need to be cropped out.

MPlayer provides a crop detection filter that will determine the crop rectangle (`-vf cropdetect`). Run MPlayer with `-vf cropdetect` and it will print out the crop settings to remove the borders. You should let the movie run long enough that the whole picture area is used, in order to get accurate crop values.

Then, test the values you get with MPlayer, using the command line which was printed by `cropdetect`, and adjust the rectangle as needed. The `rectangle` filter can help by allowing you to interactively position the crop rectangle over your movie. Remember to follow the above divisibility guidelines so that you do not misalign the chroma planes.

In certain cases, scaling may be undesirable. Scaling in the vertical direction is difficult with interlaced video, and if you wish to preserve the interlacing, you should usually refrain from scaling. If you will not be scaling but you still want to use multiple-of-16 dimensions, you will have to overcrop. Do not undercrop, since black borders are very bad for encoding!

Because MPEG-4 uses 16x16 macroblocks, you will want to make sure that each dimension of the video you are encoding is a multiple of 16 or else you will be degrading quality, especially at lower bitrates. You can do this by rounding the width and height of the crop rectangle down to the nearest multiple of 16. As stated earlier, when cropping, you will want to increase the Y offset by half the difference of the old and the new height so that the resulting video is taken from the center of the frame. And because of the way DVD video is sampled, make sure the offset is an even number. (In fact, as a rule, never use odd values for any parameter when you are cropping and scaling video.) If you are not comfortable throwing a few extra pixels away, you might prefer to scale the video instead. We will look at this in our example below. You can actually let the `cropdetect` filter do all of the above for you, as it has an optional `round` parameter that is equal to 16 by default.

Also, be careful about "half black" pixels at the edges. Make sure you crop these out too, or else you will be wasting bits there that are better spent elsewhere.

After all is said and done, you will probably end up with video whose pixels are not quite 1.85:1 or 2.35:1, but rather something close to that. You could calculate the new aspect ratio manually, but MEncoder offers an option for `libavcodec` called `autoaspect` that will do this for you. Absolutely do not scale this video up in order to square the pixels unless you like to waste your hard disk space. Scaling should be done on playback, and the player will use the aspect stored in the AVI to determine the correct resolution. Unfortunately, not all players enforce this auto-scaling information, therefore you may still want to rescale.

11.1.5. Choosing resolution and bitrate

If you will not be encoding in constant quantizer mode, you need to select a bitrate. The concept of bitrate is quite simple. It is the (average) number of bits that will be consumed to store your movie, per second. Normally bitrate is measured in kilobits (1000 bits) per second. The size of your movie on disk is the bitrate times the length of the movie in time, plus a small amount of "overhead" (see the section on the AVI container for instance). Other parameters such as scaling, cropping, etc. will **not** alter the file size unless you change the bitrate as well!

Bitrate does **not** scale proportionally to resolution. That is to say, a 320x240 file at 200 kbit/sec will not be the same quality as the same movie at 640x480 and 800 kbit/sec! There are two reasons for this:

1. **Perceptual**: You notice MPEG artifacts more if they are scaled up bigger! Artifacts appear on the scale of blocks (8x8). Your eye will not see errors in 4800 small blocks as easily as it sees errors in 1200 large blocks (assuming you will be scaling both to fullscreen).

2. **Theoretical**: When you scale down an image but still use the same size (8x8) blocks for the frequency space transform, you move more data to the high frequency bands. Roughly speaking, each pixel contains more of the detail than it did before. So even though your scaled-down picture contains 1/4 the information in the spacial directions, it could still contain a large portion of the information in the frequency domain (assuming that the high frequencies were underutilized in the original 640x480 image).

Past guides have recommended choosing a bitrate and resolution based on a "bits per pixel" approach, but this is usually not valid due to the above reasons. A better estimate seems to be that bitrates scale proportional to the square root of resolution, so that 320x240 and 400 kbit/sec would be comparable to 640x480 at 800 kbit/sec. However this has not been verified with theoretical or empirical rigor. Further, given that movies vary greatly with regard to noise, detail, degree of motion, etc., it is futile to make general recommendations for bits per length-of-diagonal (the analog of bits per pixel, using the square root).

So far we have discussed the difficulty of choosing a bitrate and resolution.

11.1.5.1. Computing the resolution

The following steps will guide you in computing the resolution of your encode without distorting the video too much, by taking into account several types of information about the source video. First, you should compute the encoded aspect ratio: `ARc = (Wc x (ARa / PRdvd )) / Hc`

where:

- Wc and Hc are the width and height of the cropped video,
- ARa is the displayed aspect ratio, which usually is 4/3 or 16/9,
- PRdvd is the pixel ratio of the DVD which is equal to 1.25=(720/576) for PAL DVDs and 1.5=(720/480) for NTSC DVDs.

Then, you can compute the X and Y resolution, according to a certain Compression Quality (CQ) factor: `ResY = INT (SQRT( 1000*Bitrate/25/ARc/CQ )/16) * 16` and `ResX = INT( ResY * ARc / 16) * 16`

Okay, but what is the CQ? The CQ represents the number of bits per pixel and per frame of the encode. Roughly speaking, the greater the CQ, the less the likelihood to see encoding artifacts. However, if you have a target size for your movie (1 or 2 CDs for instance), there is a limited total number of bits that you can spend; therefore it is necessary to find a good tradeoff between compressibility and quality.

The CQ depends on the bitrate, the video codec efficiency and the movie resolution. In order to raise the CQ, typically you would downscale the movie given that the bitrate is computed in function of the target size and the length of the movie, which are constant. With MPEG-4 ASP codecs such as `Xvid` and `libavcodec`, a CQ below 0.18 usually results in a pretty blocky picture, because there are not enough bits to code the information of each macroblock. (MPEG4, like many other codecs, groups pixels by blocks of several pixels to compress the image; if there are not enough bits, the edges of those blocks are visible.) It is therefore wise to take a CQ ranging from 0.20 to 0.22 for a 1 CD rip, and 0.26-0.28 for 2 CDs rip with standard encoding options. More advanced encoding options such as those listed here for `libavcodec` and `Xvid` should make it possible to get the same quality with CQ ranging from 0.18 to 0.20 for a 1 CD rip, and 0.24 to 0.26 for a 2 CD rip. With MPEG-4 AVC codecs such as `x264`, you can use a CQ ranging from 0.14 to 0.16 with standard encoding options, and should be able to go as low as 0.10 to 0.12 with x264's advanced encoding settings.

Please take note that the CQ is just an indicative figure, as depending on the encoded content, a CQ of 0.18 may look just fine for a Bergman, contrary to a movie such as The Matrix, which contains many high-motion scenes. On the other hand, it is worthless to raise CQ higher than 0.30 as you would be wasting bits without any noticeable quality gain. Also note that as mentioned earlier in this guide, low resolution videos need a bigger CQ (compared to, for instance, DVD resolution) to look good.

11.1.6. Filtering

Learning how to use MEncoder's video filters is essential to producing good encodes. All video processing is performed through the filters -- cropping, scaling, color adjustment, noise removal, sharpening, deinterlacing, telecine, inverse telecine, and deblocking, just to name a few. Along with the vast number of supported input formats, the variety of filters available in MEncoder is one of its main advantages over other similar programs.

Filters are loaded in a chain using the -vf option:

```
 -vf filter1=options,filter2=options,...
```

Most filters take several numeric options separated by colons, but the syntax for options varies from filter to filter, so read the man page for details on the filters you wish to use.

Filters operate on the video in the order they are loaded. For example, the following chain:

```
 -vf crop=688:464:12:4,scale=640:464
```

will first crop the 688x464 region of the picture with upper-left corner at (12,4), and then scale the result down to 640x464.

Certain filters need to be loaded at or near the beginning of the filter chain, in order to take advantage of information from the video decoder that will be lost or invalidated by other filters. The principal examples are `pp` (postprocessing, only when it is performing deblock or dering operations), `spp` (another postprocessor to remove MPEG artifacts), `pullup` (inverse telecine), and `softpulldown` (for converting soft telecine to hard telecine).

In general, you want to do as little filtering as possible to the movie in order to remain close to the original DVD source. Cropping is often necessary (as described above), but avoid to scale the video. Although scaling down is sometimes preferred to using higher quantizers, we want to avoid both these things: remember that we decided from the start to trade bits for quality.

Also, do not adjust gamma, contrast, brightness, etc. What looks good on your display may not look good on others. These adjustments should be done on playback only.

One thing you might want to do, however, is pass the video through a very light denoise filter, such as `-vf hqdn3d=2:1:2`. Again, it is a matter of putting those bits to better use: why waste them encoding noise when you can just add that noise back in during playback? Increasing the parameters for `hqdn3d` will further improve compressibility, but if you increase the values too much, you risk degrading the image visibly. The suggested values above (`2:1:2`) are quite conservative; you should feel free to experiment with higher values and observe the results for yourself.

11.1.7. Interlacing and Telecine

Almost all movies are shot at 24 fps. Because NTSC is 30000/1001 fps, some processing must be done to this 24 fps video to make it run at the correct NTSC framerate. The process is called 3:2 pulldown, commonly referred to as telecine (because pulldown is often applied during the telecine process), and, naively described, it works by slowing the film down to 24000/1001 fps, and repeating every fourth frame.

No special processing, however, is done to the video for PAL DVDs, which run at 25 fps. (Technically, PAL can be telecined, called 2:2 pulldown, but this does not become an issue in practice.) The 24 fps film is simply played back at 25 fps. The result is that the movie runs slightly faster, but unless you are an alien, you probably will not notice the difference. Most PAL DVDs have pitch-corrected audio, so when they are played back at 25 fps things will sound right, even though the audio track (and hence the whole movie) has a running time that is 4% less than NTSC DVDs.

Because the video in a PAL DVD has not been altered, you need not worry much about framerate. The source is 25 fps, and your rip will be 25 fps. However, if you are ripping an NTSC DVD movie, you may need to apply inverse telecine.

For movies shot at 24 fps, the video on the NTSC DVD is either telecined 30000/1001, or else it is progressive 24000/1001 fps and intended to be telecined on-the-fly by a DVD player. On the other hand, TV series are usually only interlaced, not telecined. This is not a hard rule: some TV series are interlaced (such as Buffy the Vampire Slayer) whereas some are a mixture of progressive and interlaced (such as Angel, or 24).

It is highly recommended that you read the section on How to deal with telecine and interlacing in NTSC DVDs to learn how to handle the different possibilities.

However, if you are mostly just ripping movies, likely you are either dealing with 24 fps progressive or telecined video, in which case you can use the `pullup` filter `-vf pullup,softskip`.

11.1.8. Encoding interlaced video

If the movie you want to encode is interlaced (NTSC video or PAL video), you will need to choose whether you want to deinterlace or not. While deinterlacing will make your movie usable on progressive scan displays such a computer monitors and projectors, it comes at a cost: The fieldrate of 50 or 60000/1001 fields per second is halved to 25 or 30000/1001 frames per second, and roughly half of the information in your movie will be lost during scenes with significant motion.

Therefore, if you are encoding for high quality archival purposes, it is recommended not to deinterlace. You can always deinterlace the movie at playback time when displaying it on progressive scan devices. The power of currently available computers forces players to use a deinterlacing filter, which results in a slight degradation in image quality. But future players will be able to mimic the interlaced display of a TV, deinterlacing to full fieldrate and interpolating 50 or 60000/1001 entire frames per second from the interlaced video.

Special care must be taken when working with interlaced video:

1. Crop height and y-offset must be multiples of 4.
2. Any vertical scaling must be performed in interlaced mode.
3. Postprocessing and denoising filters may not work as expected unless you take special care to operate them a field at a time, and they may damage the video if used incorrectly.

With these things in mind, here is our first example:

```
mencoder capture.avi -mc 0 -oac lavc -ovc lavc -lavcopts \
    vcodec=mpeg2video:vbitrate=6000:ilme:ildct:acodec=mp2:abitrate=224
```

Note the `ilme` and `ildct` options.

11.1.9. Notes on Audio/Video synchronization

MEncoder's audio/video synchronization algorithms were designed with the intention of recovering files with broken sync. However, in some cases they can cause unnecessary skipping and duplication of frames, and possibly slight A/V desync, when used with proper input (of course, A/V sync issues apply only if you process or copy the audio track while transcoding the video, which is strongly encouraged). Therefore, you may have to switch to basic A/V sync with the `-mc 0` option, or put this in your `~/.mplayer/mencoder` config file, as long as you are only working with good sources (DVD, TV capture, high quality MPEG-4 rips, etc) and not broken ASF/RM/MOV files.

If you want to further guard against strange frame skips and duplication, you can use both `-mc 0` and `-noskip`. This will prevent *all* A/V sync, and copy frames one-to-one, so you cannot use it if you will be using any filters that unpredictably add or drop frames, or if your input file has variable framerate! Therefore, using `-noskip` is not in general recommended.

The so-called "three-pass" audio encoding which MEncoder supports has been reported to cause A/V desync. This will definitely happen if it is used in conjunction with certain filters, therefore, it is now recommended *not* to use three-pass audio mode. This feature is only left for compatibility purposes and for expert users who understand when it is safe to use and when it is not. If you have never heard of three-pass mode before, forget that we even mentioned it!

There have also been reports of A/V desync when encoding from stdin with MEncoder. Do not do this! Always use a file or CD/DVD/etc device as input.

11.1.10. Choosing the video codec

Which video codec is best to choose depends on several factors, like size, quality, streamability, usability and popularity, some of which widely depend on personal taste and technical constraints.

- **Compression efficiency**: It is quite easy to understand that most newer-generation codecs are made to increase quality and compression. Therefore, the authors of this guide and many other people suggest that you cannot go wrong [1] when choosing MPEG-4 AVC codecs like `x264` instead of MPEG-4 ASP codecs such as `libavcodec` MPEG-4 or `Xvid`. (Advanced codec developers may be interested in reading Michael Niedermayer's opinion on "why MPEG4-ASP sucks".) Likewise, you should get better quality using MPEG-4 ASP than you would with MPEG-2 codecs.

  However, newer codecs which are in heavy development can suffer from bugs which have not yet been noticed and which can ruin an encode. This is simply the tradeoff for using bleeding-edge technology.

  What is more, beginning to use a new codec requires that you spend some time becoming familiar with its options, so that you know what to adjust to achieve a desired picture quality.

- **Hardware compatibility**: It usually takes a long time for standalone video players to begin to include support for the latest video codecs. As a result, most only support MPEG-1 (like VCD, XVCD and KVCD), MPEG-2 (like DVD, SVCD and KVCD) and MPEG-4 ASP (like DivX, `libavcodec`'s LMP4 and `Xvid`) (Beware: Usually, not all MPEG-4 ASP features are supported). Please refer to the technical specs of your player (if they are available), or google around for more information.

- **Best quality per encoding time**: Codecs that have been around for some time (such as `libavcodec` MPEG-4 and `Xvid`) are usually heavily optimized with all kinds of smart algorithms and SIMD assembly code. That is why they tend to yield the best quality per encoding time ratio. However, they may have some very advanced options that, if enabled, will make the encode really slow for marginal gains.

  If you are after blazing speed you should stick around the default settings of the video codec (although you should still try the other options which are mentioned in other sections of this guide).

  You may also consider choosing a codec which can do multi-threaded processing, though this is only useful for users of machines with several CPUs. libavcodec MPEG-4 does allow that, but speed gains are limited, and there is a slight negative effect on picture quality. Xvid's multi-threaded encoding, activated by the threads option, can be used to boost encoding speed — by about 40-60% in typical cases — with little if any picture degradation. x264 also allows multi-threaded encoding, which currently speeds up encoding by 94% per CPU core while lowering PSNR between 0.005dB and 0.01dB on a typical setup.

- **Personal taste**: This is where it gets almost irrational: For the same reason that some hung on to DivX 3 for years when newer codecs were already doing wonders, some folks will prefer Xvid or libavcodec MPEG-4 over x264.

  You should make your own judgement; do not take advice from people who swear by one codec. Take a few sample clips from raw sources and compare different encoding options and codecs to find one that suits you best. The best codec is the one you master, and the one that looks best to your eyes on your display [2]!

Please refer to the section selecting codecs and container formats to get a list of supported codecs.

11.1.11. Audio

Audio is a much simpler problem to solve: if you care about quality, just leave it as is. Even AC-3 5.1 streams are at most 448Kbit/s, and they are worth every bit. You might be tempted to transcode the audio to high quality Vorbis, but just because you do not have an A/V receiver for AC-3 pass-through today does not mean you will not have one tomorrow. Future-proof your DVD rips by preserving the AC-3 stream. You can keep the AC-3 stream either by copying it directly into the video stream during the encoding. You can also extract the AC-3 stream in order to mux it into containers such as NUT or Matroska.

```
mplayer source_file.vob -aid 129 -dumpaudio -dumpfile sound.ac3
```

will dump into the file *sound.ac3* the audio track number 129 from the file *source_file.vob* (NB: DVD VOB files usually use a different audio numbering, which means that the VOB audio track 129 is the 2nd audio track of the file).

But sometimes you truly have no choice but to further compress the sound so that more bits can be spent on the video. Most people choose to compress audio with either MP3 or Vorbis audio codecs. While the latter is a very space-efficient codec, MP3 is better supported by hardware players, although this trend is changing.

Do *not* use -nosound when encoding a file with audio, even if you will be encoding and muxing audio separately later. Though it may work in ideal cases, using -nosound is likely to hide some problems in your encoding command line setting. In other words, having a soundtrack during your encode assures you that, provided you do not see messages such as “Too many audio packets in the buffer”, you will be able to get proper sync.

You need to have MEncoder process the sound. You can for example copy the original soundtrack during the encode with -oac copy or convert it to a "light" 4 kHz mono WAV PCM with -oac pcm -channels 1 -srate 4000. Otherwise, in some cases, it will generate a video file that will not sync with the audio. Such cases are when the number of video frames in the source file does not match up to the total length of audio frames or whenever there are discontinuities/splices where there are missing or extra audio frames. The correct way to handle this kind of problem is to insert silence or cut audio at these points. However MPlayer cannot do that, so if you demux the AC-3 audio and encode it with a separate app (or dump it to PCM with MPlayer), the splices will be left incorrect and the only way to correct them is to drop/duplicate video frames at the splice. As long as MEncoder sees the audio when it is encoding the video, it can do this dropping/duping (which is usually OK since it takes place at full black/scene change), but if MEncoder cannot see the audio, it will just process all frames as-is and they will not fit the final audio stream when you for example merge your audio and video track into a Matroska file.

First of all, you will have to convert the DVD sound into a WAV file that the audio codec can use as input. For example:

```
mplayer source_file.vob -ao pcm:file=destination_sound.wav \
    -vc dummy -aid 1 -vo null
```

will dump the second audio track from the file *source_file.vob* into the file *destination_sound.wav*. You may want to normalize the sound before encoding, as DVD audio tracks are commonly recorded at low volumes. You can use the tool normalize for instance, which is available in most distributions. If you are using Windows, a tool such as BeSweet can do the same job. You will compress in either Vorbis or MP3. For example:

```
oggenc -q1 destination_sound.wav
```

will encode *destination_sound.wav* with the encoding quality 1, which is roughly equivalent to 80Kb/s, and is the minimum quality at which you should encode if you care about quality. Please note that MEncoder currently cannot mux Vorbis audio tracks into the output file because it only supports AVI and MPEG containers as an output, each of which may lead to audio/video playback synchronization problems with some players when the AVI file contain VBR audio streams such as Vorbis. Do not worry, this document will show you how you can do that with third party programs.

11.1.12. Muxing

Now that you have encoded your video, you will most likely want to mux it with one or more audio tracks into a movie container, such as AVI, MPEG, Matroska or NUT. MEncoder is currently only able to natively output audio and video into MPEG and AVI container formats. for example:

```
mencoder -oac copy -ovc copy  -o output_movie.avi \
    -audiofile input_audio.mp2 input_video.avi
```

This would merge the video file `input_video.avi` and the audio file `input_audio.mp2` into the AVI file `output_movie.avi`. This command works with MPEG-1 layer I, II and III (more commonly known as MP3) audio, WAV and a few other audio formats too.

MEncoder features experimental support for `libavformat`, which is a library from the FFmpeg project that supports muxing and demuxing a variety of containers. For example:

```
mencoder -oac copy -ovc copy -o output_movie.asf -audiofile input_audio.mp2 \
    input_video.avi -of lavf -lavfopts format=asf
```

This will do the same thing as the previous example, except that the output container will be ASF. Please note that this support is highly experimental (but getting better every day), and will only work if you compiled MPlayer with the support for `libavformat` enabled (which means that a pre-packaged binary version will not work in most cases).

11.1.12.1. Improving muxing and A/V sync reliability

You may experience some serious A/V sync problems while trying to mux your video and some audio tracks, where no matter how you adjust the audio delay, you will never get proper sync. That may happen when you use some video filters that will drop or duplicate some frames, like the inverse telecine filters. It is strongly encouraged to append the `harddup` video filter at the end of the filter chain to avoid this kind of problem.

Without `harddup`, if MEncoder wants to duplicate a frame, it relies on the muxer to put a mark on the container so that the last frame will be displayed again to maintain sync while writing no actual frame. With `harddup`, MEncoder will instead just push the last frame displayed again into the filter chain. This means that the encoder receives the *exact* same frame twice, and compresses it. This will result in a slightly bigger file, but will not cause problems when demuxing or remuxing into other container formats.

You may also have no choice but to use `harddup` with container formats that are not too tightly linked with MEncoder such as the ones supported through `libavformat`, which may not support frame duplication at the container level.

11.1.12.2. Limitations of the AVI container

Although it is the most widely-supported container format after MPEG-1, AVI also has some major drawbacks. Perhaps the most obvious is the overhead. For each chunk of the AVI file, 24 bytes are wasted on headers and index. This translates into a little over 5 MB per hour, or 1-2.5% overhead for a 700 MB movie. This may not seem like much, but it could mean the difference between being able to use 700 kbit/sec video or 714 kbit/sec, and every bit of quality counts.

In addition this gross inefficiency, AVI also has the following major limitations:

1. Only fixed-fps content can be stored. This is particularly limiting if the original material you want to encode is mixed content, for example a mix of NTSC video and film material. Actually there are hacks that can be used to store mixed-framerate content in AVI, but they increase the (already huge) overhead fivefold or more and so are not practical.

2. Audio in AVI files must be either constant-bitrate (CBR) or constant-framesize (i.e. all frames decode to the same number of samples). Unfortunately, the most efficient codec, Vorbis, does not meet either of these requirements.

Therefore, if you plan to store your movie in AVI, you will have to use a less efficient codec such as MP3 or AC-3.

Having said all that, MEncoder does not currently support variable-fps output or Vorbis encoding. Therefore, you may not see these as limitations if MEncoder is the only tool you will be using to produce your encodes. However, it is possible to use MEncoder only for video encoding, and then use external tools to encode audio and mux it into another container format.

11.1.12.3. Muxing into the Matroska container

Matroska is a free, open standard container format, aiming to offer a lot of advanced features, which older containers like AVI cannot handle. For example, Matroska supports variable bitrate audio content (VBR), variable framerates (VFR), chapters, file attachments, error detection code (EDC) and modern A/V Codecs like "Advanced Audio Coding" (AAC), "Vorbis" or "MPEG-4 AVC" (H.264), next to nothing handled by AVI.

The tools required to create Matroska files are collectively called mkvtoolnix, and are available for most Unix platforms as well as Windows. Because Matroska is an open standard you may find other tools that suit you better, but since mkvtoolnix is the most common, and is supported by the Matroska team itself, we will only cover its usage.

Probably the easiest way to get started with Matroska is to use MMG, the graphical frontend shipped with mkvtoolnix, and follow the guide to mkvmerge GUI (mmg)

You may also mux audio and video files using the command line:

```
mkvmerge -o output.mkv input_video.avi input_audio1.mp3 input_audio2.ac3
```

This would merge the video file *input_video.avi* and the two audio files *input_audio1.mp3* and *input_audio2.ac3* into the Matroska file *output.mkv*. Matroska, as mentioned earlier, is able to do much more than that, like multiple audio tracks (including fine-tuning of audio/video synchronization), chapters, subtitles, splitting, etc... Please refer to the documentation of those applications for more details.

11.2. How to deal with telecine and interlacing within NTSC DVDs

11.2.1. Introduction

What is telecine? If you do not understand much of what is written in this document, read the Wikipedia entry on telecine. It is an understandable and reasonably comprehensive description of what telecine is.

A note about the numbers. Many documents, including the article linked above, refer to the fields per second value of NTSC video as 59.94 and the corresponding frames per second values as 29.97 (for telecined and interlaced) and 23.976 (for progressive). For simplicity, some documents even round these numbers to 60, 30, and 24.

Strictly speaking, all those numbers are approximations. Black and white NTSC video was exactly 60 fields per second, but 60000/1001 was later chosen to accommodate color data while remaining compatible with contemporary black and white televisions. Digital NTSC video (such as on a DVD) is also 60000/1001 fields per second. From this, interlaced and telecined video are derived to be 30000/1001 frames per second; progressive video is 24000/1001 frames per second.

Older versions of the MEncoder documentation and many archived mailing list posts refer to 59.94, 29.97, and 23.976. All MEncoder documentation has been updated to use the fractional values, and you should use them too.

`-ofps 23.976` is incorrect. `-ofps 24000/1001` should be used instead.

How telecine is used. All video intended to be displayed on an NTSC television set must be 60000/1001 fields per second. Made-for-TV movies and shows are often filmed directly at 60000/1001 fields per second, but the majority of cinema is filmed at 24 or 24000/1001 frames per second. When cinematic movie DVDs are mastered, the video is then converted for television using a process called telecine.

On a DVD, the video is never actually stored as 60000/1001 fields per second. For video that was originally 60000/1001, each pair of fields is combined to form a frame, resulting in 30000/1001 frames per second. Hardware DVD players then read a flag embedded in the video stream to determine whether the odd- or even-numbered lines should form the first field.

Usually, 24000/1001 frames per second content stays as it is when encoded for a DVD, and the DVD player must perform telecining on-the-fly. Sometimes, however, the video is telecined *before* being stored on the DVD; even though it was originally 24000/1001 frames per second, it becomes 60000/1001 fields per second. When it is stored on the DVD, pairs of fields are combined to form 30000/1001 frames per second.

When looking at individual frames formed from 60000/1001 fields per second video, telecined or otherwise, interlacing is clearly visible wherever there is any motion, because one field (say, the even-numbered lines) represents a moment in time 1/(60000/1001) seconds later than the other. Playing interlaced video on a computer looks ugly both because the monitor is higher resolution and because the video is shown frame-after-frame instead of field-after-field.

Notes:

- This section only applies to NTSC DVDs, and not PAL.
- The example MEncoder lines throughout the document are **not** intended for actual use. They are simply the bare minimum required to encode the pertaining video category. How to make good DVD rips or fine-tune `libavcodec` for maximal quality is not within the scope of this section; refer to other sections within the MEncoder encoding guide.
- There are a couple footnotes specific to this guide, linked like this: [1]

11.2.2. How to tell what type of video you have

11.2.2.1. Progressive

Progressive video was originally filmed at 24000/1001 fps, and stored on the DVD without alteration.

When you play a progressive DVD in MPlayer, MPlayer will print the following line as soon as the movie begins to play:

```
demux_mpg: 24000/1001 fps progressive NTSC content detected, switching framerate.
```

From this point forward, demux_mpg should never say it finds "30000/1001 fps NTSC content."

When you watch progressive video, you should never see any interlacing. Beware, however, because sometimes there is a tiny bit of telecine mixed in where you would not expect. I have encountered TV show DVDs that have one second of telecine at every scene change, or at seemingly random places. I once watched a DVD that had a progressive first half, and the second half was telecined. If you want to be *really* thorough, you can scan the entire movie:

```
mplayer dvd://1 -nosound -vo null -benchmark
```

Using `-benchmark` makes MPlayer play the movie as quickly as it possibly can; still, depending on your hardware, it can take a while. Every time demux_mpg reports a framerate change, the line immediately above will show you the time at which the change occurred.

Sometimes progressive video on DVDs is referred to as "soft-telecine" because it is intended to be telecined by the DVD player.

11.2.2.2. Telecined

Telecined video was originally filmed at 24000/1001, but was telecined *before* it was written to the DVD.

MPlayer does not (ever) report any framerate changes when it plays telecined video.

Watching a telecined video, you will see interlacing artifacts that seem to "blink": they repeatedly appear and disappear. You can look closely at this by

1. `mplayer dvd://1`
2. Seek to a part with motion.

3. Use the . key to step forward one frame at a time.

4. Look at the pattern of interlaced-looking and progressive-looking frames. If the pattern you see is PPPII,PPPII,PPPII,... then the video is telecined. If you see some other pattern, then the video may have been telecined using some non-standard method; MEncoder cannot losslessly convert non-standard telecine to progressive. If you do not see any pattern at all, then it is most likely interlaced.

Sometimes telecined video on DVDs is referred to as "hard-telecine". Since hard-telecine is already 60000/1001 fields per second, the DVD player plays the video without any manipulation.

Another way to tell if your source is telecined or not is to play the source with the `-vf pullup` and `-v` command line options to see how `pullup` matches frames. If the source is telecined, you should see on the console a 3:2 pattern with `0+.1.+2` and `0++1` alternating. This technique has the advantage that you do not need to watch the source to identify it, which could be useful if you wish to automate the encoding procedure, or to carry out said procedure remotely via a slow connection.

11.2.2.3. Interlaced

Interlaced video was originally filmed at 60000/1001 fields per second, and stored on the DVD as 30000/1001 frames per second. The interlacing effect (often called "combing") is a result of combining pairs of fields into frames. Each field is supposed to be 1/(60000/1001) seconds apart, and when they are displayed simultaneously the difference is apparent.

As with telecined video, MPlayer should not ever report any framerate changes when playing interlaced content.

When you view an interlaced video closely by frame-stepping with the . key, you will see that every single frame is interlaced.

11.2.2.4. Mixed progressive and telecine

All of a "mixed progressive and telecine" video was originally 24000/1001 frames per second, but some parts of it ended up being telecined.

When MPlayer plays this category, it will (often repeatedly) switch back and forth between "30000/1001 fps NTSC" and "24000/1001 fps progressive NTSC". Watch the bottom of MPlayer's output to see these messages.

You should check the "30000/1001 fps NTSC" sections to make sure they are actually telecine, and not just interlaced.

11.2.2.5. Mixed progressive and interlaced

In "mixed progressive and interlaced" content, progressive and interlaced video have been spliced together.

This category looks just like "mixed progressive and telecine", until you examine the 30000/1001 fps sections and see that they do not have the telecine pattern.

11.2.3. How to encode each category

As I mentioned in the beginning, example MEncoder lines below are **not** meant to actually be used; they only demonstrate the minimum parameters to properly encode each category.

11.2.3.1. Progressive

Progressive video requires no special filtering to encode. The only parameter you need to be sure to use is `-ofps 24000/1001`. Otherwise, MEncoder will try to encode at 30000/1001 fps and will duplicate frames.

```
 mencoder dvd://1 -oac copy -ovc lavc -ofps 24000/1001
```

It is often the case, however, that a video that looks progressive actually has very short parts of telecine mixed in. Unless you are sure, it is safest to treat the video as mixed progressive and telecine. The performance loss is small [3].

11.2.3.2. Telecined

Telecine can be reversed to retrieve the original 24000/1001 content, using a process called inverse-telecine. MPlayer contains several filters to accomplish this; the best filter, `pullup`, is described in the mixed progressive and telecine section.

11.2.3.3. Interlaced

For most practical cases it is not possible to retrieve a complete progressive video from interlaced content. The only way to do so without losing half of the vertical resolution is to double the framerate and try to "guess" what ought to make up the corresponding lines for each field (this has drawbacks - see method 3).

1. Encode the video in interlaced form. Normally, interlacing wreaks havoc with the encoder's ability to compress well, but `libavcodec` has two parameters specifically for dealing with storing interlaced video a bit better: `ildct` and `ilme`. Also, using `mbd=2` is strongly recommended [2] because it will encode macroblocks as non-interlaced in places where there is no motion. Note that `-ofps` is NOT needed here.

   ```
   mencoder dvd://1 -oac copy -ovc lavc -lavcopts ildct:ilme:mbd=2
   ```

2. Use a deinterlacing filter before encoding. There are several of these filters available to choose from, each with its own advantages and disadvantages. Consult `mplayer -pphelp` and `mplayer -vf help` to see what is available (grep for "deint"), read Michael Niedermayer's Deinterlacing filters comparison, and search the MPlayer mailing lists to find many discussions about the various filters. Again, the framerate is not changing, so no `-ofps`. Also, deinterlacing should be done after cropping [1] and before scaling.

   ```
   mencoder dvd://1 -oac copy -vf yadif -ovc lavc
   ```

3. Unfortunately, this option is buggy with MEncoder; it ought to work well with MEncoder G2, but that is not here yet. You might experience crashes. Anyway, the purpose of `-vf tfields` is to create a full frame out of each field, which makes the framerate 60000/1001. The advantage of this approach is that no data is ever lost; however, since each frame comes from only one field, the missing lines have to be interpolated somehow. There are no very good methods of generating the missing data, and so the result will look a bit similar to when using some deinterlacing filters. Generating the missing lines creates other issues, as well, simply because the amount of data doubles. So, higher encoding bitrates are required to maintain quality, and more CPU power is used for both encoding and decoding. tfields has several different options for how to create the missing lines of each frame. If you use this method, then Reference the manual, and chose whichever option looks best for your material. Note that when using `tfields` you **have to** specify both `-fps` and `-ofps` to be twice the framerate of your original source.

   ```
   mencoder dvd://1 -oac copy -vf tfields=2 -ovc lavc \
       -fps 60000/1001 -ofps 60000/1001
   ```

4. If you plan on downscaling dramatically, you can extract and encode only one of the two fields. Of course, you will lose half the vertical resolution, but if you plan on downscaling to at most 1/2 of the original, the loss will not matter much. The result will be a progressive 30000/1001 frames per second file. The procedure is to use `-vf field`, then crop [1] and scale appropriately. Remember that you will have to adjust the scale to compensate for the vertical resolution being halved.

   ```
   mencoder dvd://1 -oac copy -vf field=0 -ovc lavc
   ```

11.2.3.4. Mixed progressive and telecine

In order to turn mixed progressive and telecine video into entirely progressive video, the telecined parts have to be inverse-telecined. There are three ways to accomplish this, described below. Note that you should **always** inverse-telecine before any rescaling; unless you really know what you are doing, inverse-telecine before cropping, too [1]. `-ofps 24000/1001` is needed here because the output video will be 24000/1001 frames per second.

- `-vf pullup` is designed to inverse-telecine telecined material while leaving progressive data alone. In order to work properly, `pullup` **must** be followed by the `softskip` filter or else MEncoder will crash. `pullup` is, however, the cleanest and most accurate method available for encoding both telecine and "mixed progressive and telecine".

```
mencoder dvd://1 -oac copy -vf pullup,softskip
    -ovc lavc -ofps 24000/1001
```

- `-vf filmdint` is similar to `-vf pullup`: both filters attempt to match a pair of fields to form a complete frame. `filmdint` will deinterlace individual fields that it cannot match, however, whereas `pullup` will simply drop them. Also, the two filters have separate detection code, and `filmdint` may tend to match fields a bit less often. Which filter works better may depend on the input video and personal taste; feel free to experiment with fine-tuning the filters' options if you encounter problems with either one (see the man page for details). For most well-mastered input video, however, both filters work quite well, so either one is a safe choice to start with.

  ```
  mencoder dvd://1 -oac copy -vf filmdint -ovc lavc -ofps 24000/1001
  ```

- An older method is to, rather than inverse-telecine the telecined parts, telecine the non-telecined parts and then inverse-telecine the whole video. Sound confusing? softpulldown is a filter that goes through a video and makes the entire file telecined. If we follow softpulldown with either `detc` or `ivtc`, the final result will be entirely progressive. `-ofps 24000/1001` is needed.

  ```
  mencoder dvd://1 -oac copy -vf softpulldown,ivtc=1 -ovc lavc -ofps 24000/1001
  ```

11.2.3.5. Mixed progressive and interlaced

There are two options for dealing with this category, each of which is a compromise. You should decide based on the duration/location of each type.

- Treat it as progressive. The interlaced parts will look interlaced, and some of the interlaced fields will have to be dropped, resulting in a bit of uneven jumpiness. You can use a postprocessing filter if you want to, but it may slightly degrade the progressive parts.

  This option should definitely not be used if you want to eventually display the video on an interlaced device (with a TV card, for example). If you have interlaced frames in a 24000/1001 frames per second video, they will be telecined along with the progressive frames. Half of the interlaced "frames" will be displayed for three fields' duration (3/(60000/1001) seconds), resulting in a flicking "jump back in time" effect that looks quite bad. If you even attempt this, you **must** use a deinterlacing filter like `lb` or `l5`.

  It may also be a bad idea for progressive display, too. It will drop pairs of consecutive interlaced fields, resulting in a discontinuity that can be more visible than with the second method, which shows some progressive frames twice. 30000/1001 frames per second interlaced video is already a bit choppy because it really should be shown at 60000/1001 fields per second, so the duplicate frames do not stand out as much.

  Either way, it is best to consider your content and how you intend to display it. If your video is 90% progressive and you never intend to show it on a TV, you should favor a progressive approach. If it is only half progressive, you probably want to encode it as if it is all interlaced.

- Treat it as interlaced. Some frames of the progressive parts will need to be duplicated, resulting in uneven jumpiness. Again, deinterlacing filters may slightly degrade the progressive parts.

11.2.4. Footnotes

1. **About cropping:** Video data on DVDs are stored in a format called YUV 4:2:0. In YUV video, luma ("brightness") and chroma ("color") are stored separately. Because the human eye is somewhat less sensitive to color than it is to brightness, in a YUV 4:2:0 picture there is only one chroma pixel for every four luma pixels. In a progressive picture, each square of four luma pixels (two on each side) has one common chroma pixel. You must crop progressive YUV 4:2:0 to even resolutions, and use even offsets. For example, `crop=716:380:2:26` is OK but `crop=716:380:3:26` is not.

   When you are dealing with interlaced YUV 4:2:0, the situation is a bit more complicated. Instead of every four luma pixels in the *frame* sharing a chroma pixel, every four luma pixels in each *field* share a chroma pixel. When fields are interlaced to form a frame, each scanline is one pixel high. Now, instead of all four luma pixels being in a square, there are two pixels side-by-side, and the other two pixels are side-by-side two scanlines down. The two luma pixels in the intermediate scanline are from the other field, and so share a different chroma pixel with two luma pixels two scanlines away. All this confusion makes it necessary to have vertical crop dimensions and offsets be multiples of four. Horizontal can stay even.

For telecined video, I recommend that cropping take place after inverse telecining. Once the video is progressive you only need to crop by even numbers. If you really want to gain the slight speedup that cropping first may offer, you must crop vertically by multiples of four or else the inverse-telecine filter will not have proper data.

For interlaced (not telecined) video, you must always crop vertically by multiples of four unless you use `-vf field` before cropping.

2. **About encoding parameters and quality:** Just because I recommend `mbd=2` here does not mean it should not be used elsewhere. Along with `trell`, `mbd=2` is one of the two `libavcodec` options that increases quality the most, and you should always use at least those two unless the drop in encoding speed is prohibitive (e.g. realtime encoding). There are many other options to `libavcodec` that increase encoding quality (and decrease encoding speed) but that is beyond the scope of this document.

3. **About the performance of pullup:** It is safe to use `pullup` (along with `softskip`) on progressive video, and is usually a good idea unless the source has been definitively verified to be entirely progressive. The performance loss is small for most cases. On a bare-minimum encode, `pullup` causes MEncoder to be 50% slower. Adding sound processing and advanced `lavcopts` overshadows that difference, bringing the performance decrease of using `pullup` down to 2%.

11.3. Encoding with the `libavcodec` codec family

`libavcodec` provides simple encoding to a lot of interesting video and audio formats. You can encode to the following codecs (more or less up to date):

11.3.1. `libavcodec`'s video codecs

Video codec name	Description
mjpeg	Motion JPEG
ljpeg	lossless JPEG
jpegls	JPEG LS
targa	Targa image
gif	GIF image
bmp	BMP image
png	PNG image
h261	H.261
h263	H.263
h263p	H.263+
mpeg4	ISO standard MPEG-4 (DivX, Xvid compatible)
msmpeg4	pre-standard MPEG-4 variant by MS, v3 (AKA DivX3)
msmpeg4v2	pre-standard MPEG-4 by MS, v2 (used in old ASF files)
wmv1	Windows Media Video, version 1 (AKA WMV7)
wmv2	Windows Media Video, version 2 (AKA WMV8)
rv10	RealVideo 1.0
rv20	RealVideo 2.0
mpeg1video	MPEG-1 video
mpeg2video	MPEG-2 video
huffyuv	lossless compression
ffvhuff	FFmpeg modified huffyuv lossless
asv1	ASUS Video v1
asv2	ASUS Video v2
ffv1	FFmpeg's lossless video codec

svq1	Sorenson video 1
flv	Sorenson H.263 used in Flash Video
flashsv	Flash Screen Video
dvvideo	Sony Digital Video
snow	FFmpeg's experimental wavelet-based codec
zmbv	Zip Motion Blocks Video
dnxhd	AVID DNxHD

The first column contains the codec names that should be passed after the `vcodec` config, like: `-lavcopts vcodec=msmpeg4`

An example with MJPEG compression:

```
mencoder dvd://2 -o title2.avi -ovc lavc -lavcopts vcodec=mjpeg -oac copy
```

11.3.2. `libavcodec`'s audio codecs

Audio codec name	Description
ac3	Dolby Digital (AC-3)
adpcm_*	Adaptive PCM formats - see supplementary table
flac	Free Lossless Audio Codec (FLAC)
g726	G.726 ADPCM
libfaac	Advanced Audio Coding (AAC) - using FAAC
libgsm	ETSI GSM 06.10 full rate
libgsm_ms	Microsoft GSM
libmp3lame	MPEG-1 audio layer 3 (MP3) - using LAME
mp2	MPEG-1 audio layer 2 (MP2)
pcm_*	PCM formats - see supplementary table
roq_dpcm	Id Software RoQ DPCM
sonic	experimental FFmpeg lossy codec
sonicls	experimental FFmpeg lossless codec
vorbis	Vorbis
wmav1	Windows Media Audio v1
wmav2	Windows Media Audio v2

The first column contains the codec names that should be passed after the `acodec` option, like: `-lavcopts acodec=ac3`

An example with AC-3 compression:

```
mencoder dvd://2 -o title2.avi -oac lavc -lavcopts acodec=ac3 -ovc copy
```

Contrary to `libavcodec`'s video codecs, its audio codecs do not make a wise usage of the bits they are given as they lack some minimal psychoacoustic model (if at all) which most other codec implementations feature. However, note that all these audio codecs are very fast and work out-of-the-box everywhere MEncoder has been compiled with `libavcodec` (which is the case most of time), and do not depend on external libraries.

11.3.2.1. PCM/ADPCM format supplementary table

PCM/ADPCM codec name	Description
pcm_s32le	signed 32-bit little-endian
pcm_s32be	signed 32-bit big-endian

pcm_u32le	unsigned 32-bit little-endian
pcm_u32be	unsigned 32-bit big-endian
pcm_s24le	signed 24-bit little-endian
pcm_s24be	signed 24-bit big-endian
pcm_u24le	unsigned 24-bit little-endian
pcm_u24be	unsigned 24-bit big-endian
pcm_s16le	signed 16-bit little-endian
pcm_s16be	signed 16-bit big-endian
pcm_u16le	unsigned 16-bit little-endian
pcm_u16be	unsigned 16-bit big-endian
pcm_s8	signed 8-bit
pcm_u8	unsigned 8-bit
pcm_alaw	G.711 A-LAW
pcm_mulaw	G.711 μ-LAW
pcm_s24daud	signed 24-bit D-Cinema Audio format
pcm_zork	Activision Zork Nemesis
adpcm_ima_qt	Apple QuickTime
adpcm_ima_wav	Microsoft/IBM WAVE
adpcm_ima_dk3	Duck DK3
adpcm_ima_dk4	Duck DK4
adpcm_ima_ws	Westwood Studios
adpcm_ima_smjpeg	SDL Motion JPEG
adpcm_ms	Microsoft
adpcm_4xm	4X Technologies
adpcm_xa	Phillips Yellow Book CD-ROM eXtended Architecture
adpcm_ea	Electronic Arts
adpcm_ct	Creative 16->4-bit
adpcm_swf	Adobe Shockwave Flash
adpcm_yamaha	Yamaha
adpcm_sbpro_4	Creative VOC SoundBlaster Pro 8->4-bit
adpcm_sbpro_3	Creative VOC SoundBlaster Pro 8->2.6-bit
adpcm_sbpro_2	Creative VOC SoundBlaster Pro 8->2-bit
adpcm_thp	Nintendo GameCube FMV THP
adpcm_adx	Sega/CRI ADX

11.3.3. Encoding options of libavcodec

Ideally, you would probably want to be able to just tell the encoder to switch into "high quality" mode and move on. That would probably be nice, but unfortunately hard to implement as different encoding options yield different quality results depending on the source material. That is because compression depends on the visual properties of the video in question. For example, Anime and live action have very different properties and thus require different options to obtain optimum encoding. The good news is that some options should never be left out, like `mbd=2`, `trell`, and `v4mv`. See below for a detailed description of common encoding options.

Options to adjust:

- **vmax_b_frames**: 1 or 2 is good, depending on the movie. Note that if you need to have your encode be decodable by DivX5, you need to activate closed GOP support, using `libavcodec`'s `cgop` option, but you need to deactivate scene detection, which is not a good idea as it will hurt encode efficiency a bit.
- **vb_strategy=1**: helps in high-motion scenes. On some videos, vmax_b_frames may hurt quality, but vmax_b_frames=2 along with vb_strategy=1 helps.

- **dia**: motion search range. Bigger is better and slower. Negative values are a completely different scale. Good values are -1 for a fast encode, or 2-4 for slower.
- **predia**: motion search pre-pass. Not as important as dia. Good values are 1 (default) to 4. Requires preme=2 to really be useful.
- **cmp, subcmp, precmp**: Comparison function for motion estimation. Experiment with values of 0 (default), 2 (hadamard), 3 (dct), and 6 (rate distortion). 0 is fastest, and sufficient for precmp. For cmp and subcmp, 2 is good for Anime, and 3 is good for live action. 6 may or may not be slightly better, but is slow.
- **last_pred**: Number of motion predictors to take from the previous frame. 1-3 or so help at little speed cost. Higher values are slow for no extra gain.
- **cbp, mv0**: Controls the selection of macroblocks. Small speed cost for small quality gain.
- **qprd**: adaptive quantization based on the macroblock’s complexity. May help or hurt depending on the video and other options. This can cause artifacts unless you set vqmax to some reasonably small value (6 is good, maybe as low as 4); vqmin=1 should also help.
- **qns**: very slow, especially when combined with qprd. This option will make the encoder minimize noise due to compression artifacts instead of making the encoded video strictly match the source. Do not use this unless you have already tweaked everything else as far as it will go and the results still are not good enough.
- **vqcomp**: Tweak ratecontrol. What values are good depends on the movie. You can safely leave this alone if you want. Reducing vqcomp puts more bits on low-complexity scenes, increasing it puts them on high-complexity scenes (default: 0.5, range: 0-1. recommended range: 0.5-0.7).
- **vlelim, vcelim**: Sets the single coefficient elimination threshold for luminance and chroma planes. These are encoded separately in all MPEG-like algorithms. The idea behind these options is to use some good heuristics to determine when the change in a block is less than the threshold you specify, and in such a case, to just encode the block as "no change". This saves bits and perhaps speeds up encoding. vlelim=-4 and vcelim=9 seem to be good for live movies, but seem not to help with Anime; when encoding animation, you should probably leave them unchanged.
- **qpel**: Quarter pixel motion estimation. MPEG-4 uses half pixel precision for its motion search by default, therefore this option comes with an overhead as more information will be stored in the encoded file. The compression gain/loss depends on the movie, but it is usually not very effective on Anime. qpel always incurs a significant cost in CPU decode time (+25% in practice).
- **psnr**: does not affect the actual encoding, but writes a log file giving the type/size/quality of each frame, and prints a summary of PSNR (Peak Signal to Noise Ratio) at the end.

Options not recommended to play with:

- **vme**: The default is best.
- **lumi_mask, dark_mask**: Psychovisual adaptive quantization. You do not want to play with those options if you care about quality. Reasonable values may be effective in your case, but be warned this is very subjective.
- **scplx_mask**: Tries to prevent blocky artifacts, but postprocessing is better.

11.3.4. Encoding setting examples

The following settings are examples of different encoding option combinations that affect the speed vs quality tradeoff at the same target bitrate.

All the encoding settings were tested on a 720x448 @30000/1001 fps video sample, the target bitrate was 900kbps, and the machine was an AMD-64 3400+ at 2400 MHz in 64 bits mode. Each encoding setting features the measured encoding speed (in frames per second) and the PSNR loss (in dB) compared to the "very high quality" setting. Please understand that depending on your source, your machine type and development advancements, you may get very different results.

Description	Encoding options
Very high quality	`vcodec=mpeg4:mbd=2:mv0:trell:v4mv:cbp:last_pred=3:predia=2:dia=2:vmax_b_frames=2`
High quality	`vcodec=mpeg4:mbd=2:trell:v4mv:last_pred=2:dia=-1:vmax_b_frames=2:vb_strategy=1:c`
Fast	`vcodec=mpeg4:mbd=2:trell:v4mv:turbo`
Realtime	`vcodec=mpeg4:mbd=2:turbo`

11.3.5. Custom inter/intra matrices

With this feature of `libavcodec` you are able to set custom inter (I-frames/keyframes) and intra (P-frames/predicted frames) matrices. It is supported by many of the codecs: `mpeg1video` and `mpeg2video` are reported as working.

A typical usage of this feature is to set the matrices preferred by the KVCD specifications.

The **KVCD "Notch" Quantization Matrix:**

Intra:

```
 8  9 12 22 26 27 29 34
 9 10 14 26 27 29 34 37
12 14 18 27 29 34 37 38
22 26 27 31 36 37 38 40
26 27 29 36 39 38 40 48
27 29 34 37 38 40 48 58
29 34 37 38 40 48 58 69
34 37 38 40 48 58 69 79
```

Inter:

```
16 18 20 22 24 26 28 30
18 20 22 24 26 28 30 32
20 22 24 26 28 30 32 34
22 24 26 30 32 32 34 36
24 26 28 32 34 34 36 38
26 28 30 32 34 36 38 40
28 30 32 34 36 38 42 42
30 32 34 36 38 40 42 44
```

Usage:

```
mencoder input.avi -o output.avi -oac copy -ovc lavc \
    -lavcopts inter_matrix=...:intra_matrix=...
```

```
mencoder input.avi -ovc lavc -lavcopts \
vcodec=mpeg2video:intra_matrix=8,9,12,22,26,27,29,34,9,10,14,26,27,29,34,37,\
12,14,18,27,29,34,37,38,22,26,27,31,36,37,38,40,26,27,29,36,39,38,40,48,27,\
29,34,37,38,40,48,58,29,34,37,38,40,48,58,69,34,37,38,40,48,58,69,79\
:inter_matrix=16,18,20,22,24,26,28,30,18,20,22,24,26,28,30,32,20,22,24,26,\
28,30,32,34,22,24,26,30,32,32,34,36,24,26,28,32,34,34,36,38,26,28,30,32,34,\
36,38,40,28,30,32,34,36,38,42,42,30,32,34,36,38,40,42,44 -oac copy -o svcd.mpg
```

11.3.6. Example

So, you have just bought your shiny new copy of Harry Potter and the Chamber of Secrets (widescreen edition, of course), and you want to rip this DVD so that you can add it to your Home Theatre PC. This is a region 1 DVD, so it is NTSC. The example below will still apply to PAL, except you will omit `-ofps 24000/1001` (because the output framerate is the same as the input framerate), and of course the crop dimensions will be different.

After running `mplayer dvd://1`, we follow the process detailed in the section How to deal with telecine and interlacing in NTSC DVDs and discover that it is 24000/1001 fps progressive video, which means that we need not use an inverse telecine filter, such as `pullup` or `filmdint`.

Next, we want to determine the appropriate crop rectangle, so we use the cropdetect filter:

```
mplayer dvd://1 -vf cropdetect
```

Make sure you seek to a fully filled frame (such as a bright scene, past the opening credits and logos), and you will see in MPlayer's console output:

```
crop area: X: 0..719  Y: 57..419  (-vf crop=720:362:0:58)
```

We then play the movie back with this filter to test its correctness:

```
mplayer dvd://1 -vf crop=720:362:0:58
```

And we see that it looks perfectly fine. Next, we ensure the width and height are a multiple of 16. The width is fine, however the height is not. Since we did not fail 7th grade math, we know that the nearest multiple of 16 lower than 362 is 352.

We could just use `crop=720:352:0:58`, but it would be nice to take a little off the top and a little off the bottom so that we retain the center. We have shrunk the height by 10 pixels, but we do not want to increase the y-offset by 5-pixels since that is an odd number and will adversely affect quality. Instead, we will increase the y-offset by 4 pixels:

```
mplayer dvd://1 -vf crop=720:352:0:62
```

Another reason to shave pixels from both the top and the bottom is that we ensure we have eliminated any half-black pixels if they exist. Note that if your video is telecined, make sure the `pullup` filter (or whichever inverse telecine filter you decide to use) appears in the filter chain before you crop. If it is interlaced, deinterlace before cropping. (If you choose to preserve the interlaced video, then make sure your vertical crop offset is a multiple of 4.)

If you are really concerned about losing those 10 pixels, you might prefer instead to scale the dimensions down to the nearest multiple of 16. The filter chain would look like:

```
-vf crop=720:362:0:58,scale=720:352
```

Scaling the video down like this will mean that some small amount of detail is lost, though it probably will not be perceptible. Scaling up will result in lower quality (unless you increase the bitrate). Cropping discards those pixels altogether. It is a tradeoff that you will want to consider for each circumstance. For example, if the DVD video was made for television, you might want to avoid vertical scaling, since the line sampling corresponds to the way the content was originally recorded.

On inspection, we see that our movie has a fair bit of action and high amounts of detail, so we pick 2400Kbit for our bitrate.

We are now ready to do the two pass encode. Pass one:

```
mencoder dvd://1 -ofps 24000/1001 -oac copy -o Harry_Potter_2.avi -ovc lavc \
    -lavcopts vcodec=mpeg4:vbitrate=2400:v4mv:mbd=2:trell:cmp=3:subcmp=3:autoaspect:vpass=
    -vf pullup,softskip,crop=720:352:0:62,hqdn3d=2:1:2
```

And pass two is the same, except that we specify `vpass=2`:

```
mencoder dvd://1 -ofps 24000/1001 -oac copy -o Harry_Potter_2.avi -ovc lavc \
    -lavcopts vcodec=mpeg4:vbitrate=2400:v4mv:mbd=2:trell:cmp=3:subcmp=3:autoaspect:vpass=
    -vf pullup,softskip,crop=720:352:0:62,hqdn3d=2:1:2
```

The options `v4mv:mbd=2:trell` will greatly increase the quality at the expense of encoding time. There is little reason to leave these options out when the primary goal is quality. The options `cmp=3:subcmp=3` select a comparison function that yields higher quality than the defaults. You might try experimenting with this parameter (refer to the man page for the possible values) as different functions can have a large impact on quality depending on the source material. For example, if you find `libavcodec` produces too much blocky artifacts, you could try selecting the experimental NSSE as comparison function via `*cmp=10`.

For this movie, the resulting AVI will be 138 minutes long and nearly 3GB. And because you said that file size does not matter, this is a perfectly acceptable size. However, if you had wanted it smaller, you could try a lower bitrate. Increasing bitrates have diminishing returns, so while we might clearly see an improvement from 1800Kbit to 2000Kbit, it might not be so noticeable above 2000Kbit. Feel free to experiment until you are happy.

Because we passed the source video through a denoise filter, you may want to add some of it back during playback. This, along with the `spp` post-processing filter, drastically improves the perception of quality and helps eliminate blocky artifacts in the video. With MPlayer's `autoq` option, you can vary the amount of post-processing done by the spp filter depending on available CPU. Also, at this point, you may want to apply gamma and/or color correction to best suit your display. For example:

```
mplayer Harry_Potter_2.avi -vf spp,noise=9ah:5ah,eq2=1.2 -autoq 3
```

11.4. Encoding with the `Xvid` codec

`Xvid` is a free library for encoding MPEG-4 ASP video streams. Before starting to encode, you need to set up MEncoder to support it.

This guide mainly aims at featuring the same kind of information as x264's encoding guide. Therefore, please begin by reading the first part of that guide.

11.4.1. What options should I use to get the best results?

Please begin by reviewing the `Xvid` section of MPlayer's man page. This section is intended to be a supplement to the man page.

The Xvid default settings are already a good tradeoff between speed and quality, therefore you can safely stick to them if the following section puzzles you.

11.4.2. Encoding options of `Xvid`

- **vhq** This setting affects the macroblock decision algorithm, where the higher the setting, the wiser the decision. The default setting may be safely used for every encode, while higher settings always help PSNR but are significantly slower. Please note that a better PSNR does not necessarily mean that the picture will look better, but tells you that it is closer to the original. Turning it off will noticeably speed up encoding; if speed is critical for you, the tradeoff may be worth it.

- **bvhq** This does the same job as vhq, but does it on B-frames. It has a negligible impact on speed, and slightly improves quality (around +0.1dB PSNR).

- **max_bframes** A higher number of consecutive allowed B-frames usually improves compressibility, although it may also lead to more blocking artifacts. The default setting is a good tradeoff between compressibility and quality, but you may increase it up to 3 if you are bitrate-starved. You may also decrease it to 1 or 0 if you are aiming at perfect quality, though in that case you should make sure your target bitrate is high enough to ensure that the encoder does not have to increase quantizers to reach it.

- **bf_threshold** This controls the B-frame sensitivity of the encoder, where a higher value leads to more B-frames being used (and vice versa). This setting is to be used together with `max_bframes`; if you are bitrate-starved, you should increase both `max_bframes` and `bf_threshold`, while you may increase `max_bframes` and reduce `bf_threshold` so that the encoder may use more B-frames in places that only **really** need them. A low number of `max_bframes` and a high value of `bf_threshold` is probably not a wise choice as it will force the encoder to put B-frames in places that would not benefit from them, therefore reducing visual quality. However, if you need to be compatible with standalone players that only support old DivX profiles (which only supports up to 1 consecutive B-frame), this would be your only way to increase compressibility through using B-frames.

- **trellis** Optimizes the quantization process to get an optimal tradeoff between PSNR and bitrate, which allows significant bit saving. These bits will in return be spent elsewhere on the video, raising overall visual quality. You should always leave it on as its impact on quality is huge. Even if you are looking for speed, do not disable it until you have turned down vhq and all other more CPU-hungry options to the minimum.

- **hq_ac** Activates a better coefficient cost estimation method, which slightly reduces file size by around 0.15 to 0.19% (which corresponds to less than 0.01dB PSNR increase), while having a negligible impact on speed. It is therefore recommended to always leave it on.

- **cartoon** Designed to better encode cartoon content, and has no impact on speed as it just tunes the mode decision heuristics for this type of content.

- **me_quality** This setting is to control the precision of the motion estimation. The higher `me_quality`, the more precise the estimation of the original motion will be, and the better the resulting clip will capture the original motion.

  The default setting is best in all cases; thus it is not recommended to turn it down unless you are really looking for speed, as all the bits saved by a good motion estimation would be spent elsewhere, raising overall quality. Therefore, do not go any lower than 5, and even that only as a last resort.

- **chroma_me** Improves motion estimation by also taking the chroma (color) information into account, whereas `me_quality` alone only uses luma (grayscale). This slows down encoding by 5-10% but improves visual quality quite a bit by reducing blocking effects and reduces file size by around 1.3%. If you are looking for speed, you should disable this option before starting to consider reducing `me_quality`.

- **chroma_opt** Is intended to increase chroma image quality around pure white/black edges, rather than improving compression. This can help to reduce the "red stairs" effect.

- **lumi_mask** Tries to give less bitrate to part of the picture that the human eye cannot see very well, which should allow the encoder to spend the saved bits on more important parts of the picture. The quality of the encode yielded by this option highly depends on personal preferences and on the type and monitor settings used to watch it (typically, it will not look as good if it is bright or if it is a TFT monitor).

- **qpel** Raise the number of candidate motion vectors by increasing the precision of the motion estimation from halfpel to quarterpel. The idea is to find better motion vectors which will in return reduce bitrate (hence increasing quality). However, motion vectors with quarterpel precision require a few extra bits to code, but the candidate vectors do not always give (much) better results. Quite often, the codec still spends bits on the extra precision, but little or no extra quality is gained in return. Unfortunately, there is no way to foresee the possible gains of `qpel`, so you need to actually encode with and without it to know for sure.

  `qpel` can be almost double encoding time, and requires as much as 25% more processing power to decode. It is not supported by all standalone players.

- **gmc** Tries to save bits on panning scenes by using a single motion vector for the whole frame. This almost always raises PSNR, but significantly slows down encoding (as well as decoding). Therefore, you should only use it when you have turned `vhq` to the maximum. `Xvid`'s GMC is more sophisticated than DivX's, but is only supported by few standalone players.

11.4.3. Encoding profiles

Xvid supports encoding profiles through the `profile` option, which are used to impose restrictions on the properties of the Xvid video stream such that it will be playable on anything which supports the chosen profile. The restrictions relate to resolutions, bitrates and certain MPEG-4 features. The following table shows what each profile supports.

	Simple				Advanced Simple						DivX					
Profile name	0	1	2	3	0	1	2	3	4	5	Handheld	Portable NTSC	Portable PAL	Home Theater NTSC	Home Theater PAL	HDTV
Width [pixels]	176	176	352	352	176	176	352	352	352	720	176	352	352	720	720	1280
Height [pixels]	144	144	288	288	144	144	288	288	576	576	144	240	288	480	576	720
Frame rate [fps]	15	15	15	15	30	30	15	30	30	30	15	30	25	30	25	30
Max average bitrate [kbps]	64	64	128	384	128	128	384	768	3000	8000	537.6	4854	4854	4854	4854	9708.4
Peak average																

bitrate over 3 secs [kbps]											800	8000	8000	8000	8000	16000
Max. B-frames	0	0	0	0							0	1	1	1	1	2
MPEG quantization					X	X	X	X	X	X						
Adaptive quantization					X	X	X	X	X	X	X	X	X	X	X	X
Interlaced encoding					X	X	X	X	X	X				X	X	X
Quarterpixel					X	X	X	X	X	X						
Global motion compensation					X	X	X	X	X	X						

11.4.4. Encoding setting examples

The following settings are examples of different encoding option combinations that affect the speed vs quality tradeoff at the same target bitrate.

All the encoding settings were tested on a 720x448 @30000/1001 fps video sample, the target bitrate was 900kbps, and the machine was an AMD-64 3400+ at 2400 MHz in 64 bits mode. Each encoding setting features the measured encoding speed (in frames per second) and the PSNR loss (in dB) compared to the "very high quality" setting. Please understand that depending on your source, your machine type and development advancements, you may get very different results.

Description	Encoding options	speed (in fps)	Relative PSNR loss (in dB)
Very high quality	`chroma_opt:vhq=4:bvhq=1:quant_type=mpeg`	16fps	0dB
High quality	`vhq=2:bvhq=1:chroma_opt:quant_type=mpeg`	18fps	-0.1dB
Fast	`turbo:vhq=0`	28fps	-0.69dB
Realtime	`turbo:nochroma_me:notrellis:max_bframes=0:vhq=0`	38fps	-1.48dB

11.5. Encoding with the `x264` codec

x264 is a free library for encoding H.264/AVC video streams. Before starting to encode, you need to set up MEncoder to support it.

11.5.1. Encoding options of x264

Please begin by reviewing the x264 section of MPlayer's man page. This section is intended to be a supplement to the man page. Here you will find quick hints about which options are most likely to interest most people. The man page is more terse, but also more exhaustive, and it sometimes offers much better technical detail.

11.5.1.1. Introduction

This guide considers two major categories of encoding options:

1. Options which mainly trade off encoding time vs. quality
2. Options which may be useful for fulfilling various personal preferences and special requirements

Ultimately, only you can decide which options are best for your purposes. The decision for the first class of options is the simplest: you only have to decide whether you think the quality differences justify the speed differences. For the second class of options, preferences may be far more subjective, and more factors may be involved. Note that some of the "personal preferences and special requirements" options can still have large impacts on speed or quality, but that is not what they are primarily useful for. A couple of the "personal preference" options may even cause changes that look better to some people, but look worse to others.

Before continuing, you need to understand that this guide uses only one quality metric: global PSNR. For a brief explanation of what PSNR is, see the Wikipedia article on PSNR. Global PSNR is the last PSNR number reported when you include the `psnr` option in `x264encopts`. Any time you read a claim about PSNR, one of the assumptions behind the claim is that equal bitrates are used.

Nearly all of this guide's comments assume you are using two pass. When comparing options, there are two major reasons for using two pass encoding. First, using two pass often gains around 1dB PSNR, which is a very big difference. Secondly, testing options by doing direct quality comparisons with one pass encodes introduces a major confounding factor: bitrate often varies significantly with each encode. It is not always easy to tell whether quality changes are due mainly to changed options, or if they mostly reflect essentially random differences in the achieved bitrate.

11.5.1.2. Options which primarily affect speed and quality

- **subq**: Of the options which allow you to trade off speed for quality, `subq` and `frameref` (see below) are usually by far the most important. If you are interested in tweaking either speed or quality, these are the first options you should consider. On the speed dimension, the `frameref` and `subq` options interact with each other fairly strongly. Experience shows that, with one reference frame, `subq=5` (the default setting) takes about 35% more time than `subq=1`. With 6 reference frames, the penalty grows to over 60%. `subq`'s effect on PSNR seems fairly constant regardless of the number of reference frames. Typically, `subq=5` achieves 0.2-0.5 dB higher global PSNR in comparison `subq=1`. This is usually enough to be visible.

  `subq=6` is slower and yields better quality at a reasonable cost. In comparison to `subq=5`, it usually gains 0.1-0.4 dB global PSNR with speed costs varying from 25%-100%. Unlike other levels of `subq`, the behavior of `subq=6` does not depend much on `frameref` and `me`. Instead, the effectiveness of `subq=6` depends mostly upon the number of B-frames used. In normal usage, this means `subq=6` has a large impact on both speed and quality in complex, high motion scenes, but it may not have much effect in low-motion scenes. Note that it is still recommended to always set `bframes` to something other than zero (see below).

  `subq=7` is the slowest, highest quality mode. In comparison to `subq=6`, it usually gains 0.01-0.05 dB global PSNR with speed costs varying from 15%-33%. Since the tradeoff encoding time vs. quality is quite low, you should only use it if you are after every bit saving and if encoding time is not an issue.

- **frameref**: `frameref` is set to 1 by default, but this should not be taken to imply that it is reasonable to set it to 1. Merely raising `frameref` to 2 gains around 0.15dB PSNR with a 5-10% speed penalty; this seems like a good tradeoff. `frameref=3` gains around 0.25dB PSNR over `frameref=1`, which should be a visible difference. `frameref=3` is around 15% slower than `frameref=1`. Unfortunately, diminishing returns set in rapidly. `frameref=6` can be expected to gain only 0.05-0.1 dB over `frameref=3` at an additional 15% speed penalty. Above `frameref=6`, the quality gains are usually very small (although you should keep in mind throughout this whole discussion that it can vary quite a lot depending on your source). In a fairly typical case, `frameref=12` will improve global PSNR by a tiny 0.02dB over `frameref=6`, at a speed cost of 15%-20%. At such high `frameref` values, the only really good thing that can be said is that increasing it even further will almost certainly never **harm** PSNR, but the additional quality benefits are barely even measurable, let alone perceptible.

  #### Note:

  Raising `frameref` to unnecessarily high values **can** and **usually does** hurt coding efficiency if you turn CABAC off. With CABAC on (the default behavior), the possibility of setting `frameref` "too high" currently seems too remote to even worry about, and in the future, optimizations may remove the possibility altogether.

  If you care about speed, a reasonable compromise is to use low `subq` and `frameref` values on the first pass, and then raise them on the second pass. Typically, this has a negligible negative effect on the final quality: You will probably lose well under 0.1dB PSNR, which should be much too small of a difference to see. However, different values of `frameref` can occasionally affect frame type decision. Most likely, these are rare outlying cases, but if you want to be pretty sure, consider whether your video has either fullscreen repetitive flashing patterns or very large temporary occlusions which might force an I-frame. Adjust the first-pass `frameref` so it is large enough to contain the duration of the flashing cycle (or occlusion). For example, if the scene flashes back and forth between two images over a duration of three frames, set the first pass `frameref` to 3 or higher. This issue is probably extremely rare in live action video material, but it does sometimes come up in video game captures.

- **me**: This option is for choosing the motion estimation search method. Altering this option provides a straightforward quality-vs-speed tradeoff. `me=dia` is only a few percent faster than the default search, at a cost of under 0.1dB global PSNR. The default setting (`me=hex`) is a reasonable tradeoff between speed and quality. `me=umh` gains a little under 0.1dB global PSNR, with a speed penalty that varies depending on `frameref`. At high values of `frameref` (e.g. 12 or so), `me=umh` is about 40% slower than the default `me=hex`. With `frameref=3`, the speed penalty incurred drops to 25%-30%.

`me=esa` uses an exhaustive search that is too slow for practical use.

- **partitions=all**: This option enables the use of 8x4, 4x8 and 4x4 subpartitions in predicted macroblocks (in addition to the default partitions). Enabling it results in a fairly consistent 10%-15% loss of speed. This option is rather useless in source containing only low motion, however in some high-motion source, particularly source with lots of small moving objects, gains of about 0.1dB can be expected.

- **bframes**: If you are used to encoding with other codecs, you may have found that B-frames are not always useful. In H.264, this has changed: there are new techniques and block types that are possible in B-frames. Usually, even a naive B-frame choice algorithm can have a significant PSNR benefit. It is interesting to note that using B-frames usually speeds up the second pass somewhat, and may also speed up a single pass encode if adaptive B-frame decision is turned off.

  With adaptive B-frame decision turned off (`x264encopts`'s `nob_adapt`), the optimal value for this setting is usually no more than `bframes=1`, or else high-motion scenes can suffer. With adaptive B-frame decision on (the default behavior), it is safe to use higher values; the encoder will reduce the use of B-frames in scenes where they would hurt compression. The encoder rarely chooses to use more than 3 or 4 B-frames; setting this option any higher will have little effect.

- **b_adapt**: Note: This is on by default.

  With this option enabled, the encoder will use a reasonably fast decision process to reduce the number of B-frames used in scenes that might not benefit from them as much. You can use `b_bias` to tweak how B-frame-happy the encoder is. The speed penalty of adaptive B-frames is currently rather modest, but so is the potential quality gain. It usually does not hurt, however. Note that this only affects speed and frame type decision on the first pass. `b_adapt` and `b_bias` have no effect on subsequent passes.

- **b_pyramid**: You might as well enable this option if you are using >=2 B-frames; as the man page says, you get a little quality improvement at no speed cost. Note that these videos cannot be read by libavcodec-based decoders older than about March 5, 2005.

- **weight_b**: In typical cases, there is not much gain with this option. However, in crossfades or fade-to-black scenes, weighted prediction gives rather large bitrate savings. In MPEG-4 ASP, a fade-to-black is usually best coded as a series of expensive I-frames; using weighted prediction in B-frames makes it possible to turn at least some of these into much smaller B-frames. Encoding time cost is minimal, as no extra decisions need to be made. Also, contrary to what some people seem to guess, the decoder CPU requirements are not much affected by weighted prediction, all else being equal.

  Unfortunately, the current adaptive B-frame decision algorithm has a strong tendency to avoid B-frames during fades. Until this changes, it may be a good idea to add `nob_adapt` to your x264encopts, if you expect fades to have a large effect in your particular video clip.

- **threads**: This option allows to spawn threads to encode in parallel on multiple CPUs. You can manually select the number of threads to be created or, better, set `threads=auto` and let `x264` detect how many CPUs are available and pick an appropriate number of threads. If you have a multi-processor machine, you should really consider using it as it can to increase encoding speed linearly with the number of CPU cores (about 94% per CPU core), with very little quality reduction (about 0.005dB for dual processor, about 0.01dB for a quad processor machine).

11.5.1.3. Options pertaining to miscellaneous preferences

- **Two pass encoding**: Above, it was suggested to always use two pass encoding, but there are still reasons for not using it. For instance, if you are capturing live TV and encoding in realtime, you are forced to use single-pass. Also, one pass is obviously faster than two passes; if you use the exact same set of options on both passes, two pass encoding is almost twice as slow.

  Still, there are very good reasons for using two pass encoding. For one thing, single pass ratecontrol is not psychic, and it often makes unreasonable choices because it cannot see the big picture. For example, suppose you have a two minute long video consisting of two distinct halves. The first half is a very high-motion scene lasting 60 seconds which, in isolation, requires about 2500kbps in order to look decent. Immediately following it is a much less demanding 60-second scene that looks good at 300kbps. Suppose you ask for 1400kbps on the theory that this is enough to accommodate both scenes. Single pass ratecontrol will make a couple of "mistakes" in such a case. First of all, it will target 1400kbps in both segments. The first segment may end up heavily overquantized, causing it to look unacceptably and unreasonably blocky. The second segment will be heavily underquantized; it may look perfect, but the bitrate cost of that perfection will be completely unreasonable. What is even harder to avoid is the problem at the transition between the two scenes. The first seconds of the low motion half will be hugely over-quantized, because the ratecontrol is still expecting the kind of bitrate requirements it met in the first half of the video. This "error period" of heavily over-quantized low motion will look jarringly bad, and will actually use less than the 300kbps it would have taken to make it look decent. There are ways to mitigate the pitfalls of single-pass encoding, but they may tend to increase bitrate misprediction.

Multipass ratecontrol can offer huge advantages over a single pass. Using the statistics gathered from the first pass encode, the encoder can estimate, with reasonable accuracy, the "cost" (in bits) of encoding any given frame, at any given quantizer. This allows for a much more rational, better planned allocation of bits between the expensive (high-motion) and cheap (low-motion) scenes. See `qcomp` below for some ideas on how to tweak this allocation to your liking.

Moreover, two passes need not take twice as long as one pass. You can tweak the options in the first pass for higher speed and lower quality. If you choose your options well, you can get a very fast first pass. The resulting quality in the second pass will be slightly lower because size prediction is less accurate, but the quality difference is normally much too small to be visible. Try, for example, adding `subq=1:frameref=1` to the first pass `x264encopts`. Then, on the second pass, use slower, higher-quality options: `subq=6:frameref=15:partitions=all:me=umh`

- **Three pass encoding**? x264 offers the ability to make an arbitrary number of consecutive passes. If you specify `pass=1` on the first pass, then use `pass=3` on a subsequent pass, the subsequent pass will both read the statistics from the previous pass, and write its own statistics. An additional pass following this one will have a very good base from which to make highly accurate predictions of frame sizes at a chosen quantizer. In practice, the overall quality gain from this is usually close to zero, and quite possibly a third pass will result in slightly worse global PSNR than the pass before it. In typical usage, three passes help if you get either bad bitrate prediction or bad looking scene transitions when using only two passes. This is somewhat likely to happen on extremely short clips. There are also a few special cases in which three (or more) passes are handy for advanced users, but for brevity, this guide omits discussing those special cases.

- **qcomp**: `qcomp` trades off the number of bits allocated to "expensive" high-motion versus "cheap" low-motion frames. At one extreme, `qcomp=0` aims for true constant bitrate. Typically this would make high-motion scenes look completely awful, while low-motion scenes would probably look absolutely perfect, but would also use many times more bitrate than they would need in order to look merely excellent. At the other extreme, `qcomp=1` achieves nearly constant quantization parameter (QP). Constant QP does not look bad, but most people think it is more reasonable to shave some bitrate off of the extremely expensive scenes (where the loss of quality is not as noticeable) and reallocate it to the scenes that are easier to encode at excellent quality. `qcomp` is set to 0.6 by default, which may be slightly low for many peoples' taste (0.7-0.8 are also commonly used).

- **keyint**: `keyint` is solely for trading off file seekability against coding efficiency. By default, `keyint` is set to 250. In 25fps material, this guarantees the ability to seek to within 10 seconds precision. If you think it would be important and useful to be able to seek within 5 seconds of precision, set `keyint=125`; this will hurt quality/bitrate slightly. If you care only about quality and not about seekability, you can set it to much higher values (understanding that there are diminishing returns which may become vanishingly low, or even zero). The video stream will still have seekable points as long as there are some scene changes.

- **deblock**: This topic is going to be a bit controversial.

  H.264 defines a simple deblocking procedure on I-blocks that uses pre-set strengths and thresholds depending on the QP of the block in question. By default, high QP blocks are filtered heavily, and low QP blocks are not deblocked at all. The pre-set strengths defined by the standard are well-chosen and the odds are very good that they are PSNR-optimal for whatever video you are trying to encode. The `deblock` allow you to specify offsets to the preset deblocking thresholds.

  Many people seem to think it is a good idea to lower the deblocking filter strength by large amounts (say, -3). This is however almost never a good idea, and in most cases, people who are doing this do not understand very well how deblocking works by default.

  The first and most important thing to know about the in-loop deblocking filter is that the default thresholds are almost always PSNR-optimal. In the rare cases that they are not optimal, the ideal offset is plus or minus 1. Adjusting deblocking parameters by a larger amount is almost guaranteed to hurt PSNR. Strengthening the filter will smear more details; weakening the filter will increase the appearance of blockiness.

It is definitely a bad idea to lower the deblocking thresholds if your source is mainly low in spacial complexity (i.e., not a lot of detail or noise). The in-loop filter does a rather excellent job of concealing the artifacts that occur. If the source is high in spacial complexity, however, artifacts are less noticeable. This is because the ringing tends to look like detail or noise. Human visual perception easily notices when detail is removed, but it does not so easily notice when the noise is wrongly represented. When it comes to subjective quality, noise and detail are somewhat interchangeable. By lowering the deblocking filter strength, you are most likely increasing error by adding ringing artifacts, but the eye does not notice because it confuses the artifacts with detail.

This **still** does not justify lowering the deblocking filter strength, however. You can generally get better quality noise from postprocessing. If your H.264 encodes look too blurry or smeared, try playing with `-vf noise` when you play your encoded movie. `-vf noise=8a:4a` should conceal most mild artifacts. It will almost certainly look better than the results you would have gotten just by fiddling with the deblocking filter.

11.5.2. Encoding setting examples

The following settings are examples of different encoding option combinations that affect the speed vs quality tradeoff at the same target bitrate.

All the encoding settings were tested on a 720x448 @30000/1001 fps video sample, the target bitrate was 900kbps, and the machine was an AMD-64 3400+ at 2400 MHz in 64 bits mode. Each encoding setting features the measured encoding speed (in frames per second) and the PSNR loss (in dB) compared to the "very high quality" setting. Please understand that depending on your source, your machine type and development advancements, you may get very different results.

Description	Encoding options	speed (in fps)
Very high quality	`subq=6:partitions=all:8x8dct:me=umh:frameref=5:bframes=3:b_pyramid:weight_b`	6fps
High quality	`subq=5:8x8dct:frameref=2:bframes=3:b_pyramid:weight_b`	13fps
Fast	`subq=4:bframes=2:b_pyramid:weight_b`	17fps

11.6. Encoding with the `Video For Windows` codec family

Video for Windows provides simple encoding by means of binary video codecs. You can encode with the following codecs (if you have more, please tell us!)

Note that support for this is very experimental and some codecs may not work correctly. Some codecs will only work in certain colorspaces, try `-vf format=bgr24` and `-vf format=yuy2` if a codec fails or gives wrong output.

11.6.1. Video for Windows supported codecs

Video codec file name	Description (FourCC)	md5sum	Comment
aslcodec_vfw.dll	Alparysoft lossless codec vfw (ASLC)	608af234a6ea4d90cdc7246af5f3f29a	
avimszh.dll	AVImszh (MSZH)	253118fe1eedea04a95ed6e5f4c28878	needs `-vf format`
avizlib.dll	AVIzlib (ZLIB)	2f1cc76bbcf6d77d40d0e23392fa8eda	
divx.dll	DivX4Windows-VFW	acf35b2fc004a89c829531555d73f1e6	
huffyuv.dll	HuffYUV (lossless) (HFYU)	b74695b50230be4a6ef2c4293a58ac3b	
iccvid.dll	Cinepak Video (cvid)	cb3b7ee47ba7dbb3d23d34e274895133	
icmw_32.dll	Motion Wavelets	c9618a8fc73ce219ba918e3e09e227f2	

	(MWV1)		
jp2avi.dll	ImagePower MJPEG2000 (IPJ2)	d860a11766da0d0ea064672c6833768b	`-vf flip`
m3jp2k32.dll	Morgan MJPEG2000 (MJ2C)	f3c174edcbaef7cb947d6357cdfde7ff	
m3jpeg32.dll	Morgan Motion JPEG Codec (MJPEG)	1cd13fff5960aa2aae43790242c323b1	
mpg4c32.dll	Microsoft MPEG-4 v1/v2	b5791ea23f33010d37ab8314681f1256	
tsccvid.dll	TechSmith Camtasia Screen Codec (TSCC)	8230d8560c41d444f249802a2700d1d5	shareware error on windows
vp31vfw.dll	On2 Open Source VP3 Codec (VP31)	845f3590ea489e2e45e876ab107ee7d2	
vp4vfw.dll	On2 VP4 Personal Codec (VP40)	fc5480a482ccc594c2898dcc4188b58f	
vp6vfw.dll	On2 VP6 Personal Codec (VP60)	04d635a364243013898fd09484f913fb	
vp7vfw.dll	On2 VP7 Personal Codec (VP70)	cb4cc3d4ea7c94a35f1d81c3d750bc8d	`-ffourcc VP70`
ViVD2.dll	SoftMedia ViVD V2 codec VfW (GXVE)	a7b4bf5cac630bb9262c3f80d8a773a1	
msulvc06.DLL	MSU Lossless codec (MSUD)	294bf9288f2f127bb86f00bfcc9ccdda	Decodable by Window Media Player, not MPlayer (yet).
camcodec.dll	CamStudio lossless video codec (CSCD)	0efe97ce08bb0e40162ab15ef3b45615	sf.net/projects/camstudio

The first column contains the codec names that should be passed after the `codec` parameter, like: `-xvfwopts codec=divx.dll` The FourCC code used by each codec is given in the parentheses.

An example to convert an ISO DVD trailer to a VP6 flash video file using compdata bitrate settings:

```
mencoder -dvd-device zeiram.iso dvd://7 -o trailer.flv \
-ovc vfw -xvfwopts codec=vp6vfw.dll:compdata=onepass.mcf -oac mp3lame \
-lameopts cbr:br=64 -af lavcresample=22050 -vf yadif,scale=320:240,flip \
-of lavf
```

11.6.2. Using vfw2menc to create a codec settings file.

To encode with the Video for Windows codecs, you will need to set bitrate and other options. This is known to work on x86 on both *NIX and Windows.

First you must build the vfw2menc program. It is located in the `TOOLS` subdirectory of the MPlayer source tree. To build on Linux, this can be done using Wine:

```
winegcc vfw2menc.c -o vfw2menc -lwinmm -lole32
```

To build on Windows in MinGW or Cygwin use:

```
gcc vfw2menc.c -o vfw2menc.exe -lwinmm -lole32
```

To build on MSVC you will need getopt. Getopt can be found in the original vfw2menc archive available at: The MPlayer on win32 project.

Below is an example with the VP6 codec.

```
vfw2menc -f VP62 -d vp6vfw.dll -s firstpass.mcf
```

This will open the VP6 codec dialog window. Repeat this step for the second pass and use `-s secondpass.mcf`.

Windows users can use `-xvfwopts codec=vp6vfw.dll:compdata=dialog` to have the codec dialog display before encoding starts.

11.7. Using MEncoder to create QuickTime-compatible files

11.7.1. Why would one want to produce QuickTime-compatible Files?

There are several reasons why producing QuickTime-compatible files can be desirable.

- You want any computer illiterate to be able to watch your encode on any major platform (Windows, Mac OS X, Unices …).
- QuickTime is able to take advantage of more hardware and software acceleration features of Mac OS X than platform-independent players like MPlayer or VLC. That means that your encodes have a chance to be played smoothly by older G4-powered machines.
- QuickTime 7 supports the next-generation codec H.264, which yields significantly better picture quality than previous codec generations (MPEG-2, MPEG-4 …).

11.7.2. QuickTime 7 limitations

QuickTime 7 supports H.264 video and AAC audio, but it does not support them muxed in the AVI container format. However, you can use MEncoder to encode the video and audio, and then use an external program such as mp4creator (part of the MPEG4IP suite) to remux the video and audio tracks into an MP4 container.

QuickTime’s support for H.264 is limited, so you will need to drop some advanced features. If you encode your video with features that QuickTime 7 does not support, QuickTime-based players will show you a pretty white screen instead of your expected video.

- **B-frames**: QuickTime 7 supports a maximum of 1 B-frame, i.e. `-x264encopts bframes=1`. This means that `b_pyramid` and `weight_b` will have no effect, since they require `bframes` to be greater than 1.
- **Macroblocks**: QuickTime 7 does not support 8x8 DCT macroblocks. This option (`8x8dct`) is off by default, so just be sure not to explicitly enable it. This also means that the `i8x8` option will have no effect, since it requires `8x8dct`.
- **Aspect ratio**: QuickTime 7 does not support SAR (sample aspect ratio) information in MPEG-4 files; it assumes that SAR=1. Read the section on scaling for a workaround. QuickTime X no longer has this limitation.

11.7.3. Cropping

Suppose you want to rip your freshly bought copy of "The Chronicles of Narnia". Your DVD is region 1, which means it is NTSC. The example below would still apply to PAL, except you would omit `-ofps 24000/1001` and use slightly different `crop` and `scale` dimensions.

After running `mplayer dvd://1`, you follow the process detailed in the section How to deal with telecine and interlacing in NTSC DVDs and discover that it is 24000/1001 fps progressive video. This simplifies the process somewhat, since you do not need to use an inverse telecine filter such as `pullup` or a deinterlacing filter such as `yadif`.

Next, you need to crop out the black bars from the top and bottom of the video, as detailed in this previous section.

11.7.4. Scaling

The next step is truly heartbreaking. QuickTime 7 does not support MPEG-4 videos with a sample aspect ratio other than 1, so you will need to upscale (which wastes a lot of disk space) or downscale (which loses some details of the source) the video to square pixels. Either way you do it, this is highly inefficient, but simply cannot be avoided if you want your video to be playable by QuickTime 7. MEncoder can apply the appropriate upscaling or downscaling by specifying respectively `-vf scale=-10:-1` or `-vf scale=-1:-10`. This will scale your video to the correct width for the cropped height, rounded to the closest multiple of 16 for optimal compression. Remember that if you are cropping, you should crop first, then scale:

```
-vf crop=720:352:0:62,scale=-10:-1
```

11.7.5. A/V sync

Because you will be remuxing into a different container, you should always use the `harddup` option to ensure that duplicated frames are actually duplicated in the video output. Without this option, MEncoder will simply put a marker in the video stream that a frame was duplicated, and rely on the client software to show the same frame twice. Unfortunately, this "soft duplication" does not survive remuxing, so the audio would slowly lose sync with the video.

The final filter chain looks like this:

```
-vf crop=720:352:0:62,scale=-10:-1,harddup
```

11.7.6. Bitrate

As always, the selection of bitrate is a matter of the technical properties of the source, as explained here, as well as a matter of taste. This movie has a fair bit of action and lots of detail, but H.264 video looks good at much lower bitrates than XviD or other MPEG-4 codecs. After much experimentation, the author of this guide chose to encode this movie at 900kbps, and thought that it looked very good. You may decrease bitrate if you need to save more space, or increase it if you need to improve quality.

11.7.7. Encoding example

You are now ready to encode the video. Since you care about quality, of course you will be doing a two-pass encode. To shave off some encoding time, you can specify the `turbo` option on the first pass; this reduces `subq` and `frameref` to 1. To save some disk space, you can use the `ss` option to strip off the first few seconds of the video. (I found that this particular movie has 32 seconds of credits and logos.) `bframes` can be 0 or 1. The other options are documented in Encoding with the `x264` codec and the man page.

```
mencoder dvd://1 -o /dev/null -ss 32 -ovc x264 \
-x264encopts pass=1:turbo:bitrate=900:bframes=1:\
me=umh:partitions=all:trellis=1:qp_step=4:qcomp=0.7:direct_pred=auto:keyint=300 \
-vf crop=720:352:0:62,scale=-10:-1,harddup \
-oac faac -faacopts br=192:mpeg=4:object=2 -channels 2 -srate 48000 \
-ofps 24000/1001
```

If you have a multi-processor machine, don't miss the opportunity to dramatically speed-up encoding by enabling `x264's` multi-threading mode by adding `threads=auto` to your `x264encopts` command-line.

The second pass is the same, except that you specify the output file and set `pass=2`.

```
mencoder dvd://1 -o narnia.avi -ss 32 -ovc x264 \
-x264encopts pass=2:turbo:bitrate=900:frameref=5:bframes=1:\
me=umh:partitions=all:trellis=1:qp_step=4:qcomp=0.7:direct_pred=auto:keyint=300 \
-vf crop=720:352:0:62,scale=-10:-1,harddup \
-oac faac -faacopts br=192:mpeg=4:object=2 -channels 2 -srate 48000 \
-ofps 24000/1001
```

The resulting AVI should play perfectly in MPlayer, but of course QuickTime can not play it because it does not support H.264 muxed in AVI. So the next step is to remux the video into an MP4 container.

11.7.8. Remuxing as MP4

There are several ways to remux AVI files to MP4. You can use mp4creator, which is part of the MPEG4IP suite.

First, demux the AVI into separate audio and video streams using MPlayer.

```
mplayer narnia.avi -dumpaudio -dumpfile narnia.aac
mplayer narnia.avi -dumpvideo -dumpfile narnia.h264
```

The file names are important; mp4creator requires that AAC audio streams be named `.aac` and H.264 video streams be named `.h264`.

Now use mp4creator to create a new MP4 file out of the audio and video streams.

```
mp4creator -create=narnia.aac narnia.mp4
mp4creator -create=narnia.h264 -rate=23.976 narnia.mp4
```

Unlike the encoding step, you must specify the framerate as a decimal (such as 23.976), not a fraction (such as 24000/1001).

This `narnia.mp4` file should now be playable with any QuickTime 7 application, such as QuickTime Player or iTunes. If you are planning to view the video in a web browser with the QuickTime plugin, you should also hint the movie so that the QuickTime plugin can start playing it while it is still downloading. mp4creator can create these hint tracks:

```
mp4creator -hint=1 narnia.mp4
mp4creator -hint=2 narnia.mp4
mp4creator -optimize narnia.mp4
```

You can check the final result to ensure that the hint tracks were created successfully:

```
mp4creator -list narnia.mp4
```

You should see a list of tracks: 1 audio, 1 video, and 2 hint tracks.

```
Track   Type    Info
1       audio   MPEG-4 AAC LC, 8548.714 secs, 190 kbps, 48000 Hz
2       video   H264 Main@5.1, 8549.132 secs, 899 kbps, 848x352 @ 23.976001 fps
3       hint    Payload mpeg4-generic for track 1
4       hint    Payload H264 for track 2
```

11.7.9. Adding metadata tags

If you want to add tags to your video that show up in iTunes, you can use AtomicParsley.

```
AtomicParsley narnia.mp4 --metaEnema --title "The Chronicles of Narnia" --year 2005 --stik
```

The `--metaEnema` option removes any existing metadata (mp4creator inserts its name in the "encoding tool" tag), and `--freefree` reclaims the space from the deleted metadata. The `--stik` option sets the type of video (such as Movie or TV Show), which iTunes uses to group related video files. The `--overWrite` option overwrites the original file; without it, AtomicParsley creates a new auto-named file in the same directory and leaves the original file untouched.

11.8. Using MEncoder to create VCD/SVCD/DVD-compliant files

11.8.1. Format Constraints

MEncoder is capable of creating VCD, SCVD and DVD format MPEG files using the `libavcodec` library. These files can then be used in conjunction with vcdimager or dvdauthor to create discs that will play on a standard set-top player.

The DVD, SVCD, and VCD formats are subject to heavy constraints. Only a small selection of encoded picture sizes and aspect ratios are available. If your movie does not already meet these requirements, you may have to scale, crop or add black borders to the picture to make it compliant.

11.8.1.1. Format Constraints

Format	Resolution	V. Codec	V. Bitrate	Sample Rate	A. Codec	A. Bitrate	FPS	Aspect
NTSC DVD	720x480, 704x480, 352x480, 352x240	MPEG-2	9800 kbps	48000 Hz	AC-3,PCM	1536 kbps (max)	30000/1001, 24000/1001	4:3, 16:9 (only for 720x480)
NTSC DVD	352x240[a]	MPEG-1	1856 kbps	48000 Hz	AC-3,PCM	1536 kbps (max)	30000/1001, 24000/1001	4:3, 16:9
NTSC SVCD	480x480	MPEG-2	2600 kbps	44100 Hz	MP2	384 kbps (max)	30000/1001	4:3
NTSC VCD	352x240	MPEG-1	1150 kbps	44100 Hz	MP2	224 kbps	24000/1001, 30000/1001	4:3
PAL DVD	720x576, 704x576, 352x576, 352x288	MPEG-2	9800 kbps	48000 Hz	MP2,AC-3,PCM	1536 kbps (max)	25	4:3, 16:9 (only for 720x576)
PAL DVD	352x288[a]	MPEG-1	1856 kbps	48000 Hz	MP2,AC-3,PCM	1536 kbps (max)	25	4:3, 16:9
PAL SVCD	480x576	MPEG-2	2600 kbps	44100 Hz	MP2	384 kbps (max)	25	4:3
PAL VCD	352x288	MPEG-1	1152 kbps	44100 Hz	MP2	224 kbps	25	4:3

[a] These resolutions are rarely used for DVDs because they are fairly low quality.

If your movie has 2.35:1 aspect (most recent action movies), you will have to add black borders or crop the movie down to 16:9 to make a DVD or VCD. If you add black borders, try to align them at 16-pixel boundaries in order to minimize the impact on encoding performance. Thankfully DVD has sufficiently excessive bitrate that you do not have to worry too much about encoding efficiency, but SVCD and VCD are highly bitrate-starved and require effort to obtain acceptable quality.

11.8.1.2. GOP Size Constraints

DVD, VCD, and SVCD also constrain you to relatively low GOP (Group of Pictures) sizes. For 30 fps material the largest allowed GOP size is 18. For 25 or 24 fps, the maximum is 15. The GOP size is set using the `keyint` option.

11.8.1.3. Bitrate Constraints

VCD video is required to be CBR at 1152 kbps. This highly limiting constraint also comes along with an extremely low vbv buffer size of 327 kilobits. SVCD allows varying video bitrates up to 2500 kbps, and a somewhat less restrictive vbv buffer size of 917 kilobits is allowed. DVD video bitrates may range anywhere up to 9800 kbps (though typical bitrates are about half that), and the vbv buffer size is 1835 kilobits.

11.8.2. Output Options

MEncoder has options to control the output format. Using these options we can instruct it to create the correct type of file.

The options for VCD and SVCD are called xvcd and xsvcd, because they are extended formats. They are not strictly compliant, mainly because the output does not contain scan offsets. If you need to generate an SVCD image, you should pass the output file to vcdimager.

VCD:

```
 -of mpeg -mpegopts format=xvcd
```

SVCD:

```
 -of mpeg -mpegopts format=xsvcd
```

DVD (with timestamps on every frame, if possible):

```
 -of mpeg -mpegopts format=dvd:tsaf
```

DVD with NTSC Pullup:

```
 -of mpeg -mpegopts format=dvd:tsaf:telecine -ofps 24000/1001
```

This allows 24000/1001 fps progressive content to be encoded at 30000/1001 fps whilst maintaining DVD-compliance.

11.8.2.1. Aspect Ratio

The aspect argument of `-lavcopts` is used to encode the aspect ratio of the file. During playback the aspect ratio is used to restore the video to the correct size.

16:9 or "Widescreen"

```
 -lavcopts aspect=16/9
```

4:3 or "Fullscreen"

```
 -lavcopts aspect=4/3
```

2.35:1 or "Cinemascope" NTSC

```
 -vf scale=720:368,expand=720:480 -lavcopts aspect=16/9
```

To calculate the correct scaling size, use the expanded NTSC width of 854/2.35 = 368

2.35:1 or "Cinemascope" PAL

```
 -vf scale=720:432,expand=720:576 -lavcopts aspect=16/9
```

To calculate the correct scaling size, use the expanded PAL width of 1024/2.35 = 432

11.8.2.2. Maintaining A/V sync

In order to maintain audio/video synchronization throughout the encode, MEncoder has to drop or duplicate frames. This works rather well when muxing into an AVI file, but is almost guaranteed to fail to maintain A/V sync with other muxers such as MPEG. This is why it is necessary to append the `harddup` video filter at the end of the filter chain to avoid this kind of problem. You can find more technical information about `harddup` in the section Improving muxing and A/V sync reliability or in the manual page.

11.8.2.3. Sample Rate Conversion

If the audio sample rate in the original file is not the same as required by the target format, sample rate conversion is required. This is achieved using the `-srate` option and the `-af lavcresample` audio filter together.

DVD:

```
 -srate 48000 -af lavcresample=48000
```

VCD and SVCD:

```
 -srate 44100 -af lavcresample=44100
```

11.8.3. Using libavcodec for VCD/SVCD/DVD Encoding

11.8.3.1. Introduction

`libavcodec` can be used to create VCD/SVCD/DVD compliant video by using the appropriate options.

11.8.3.2. lavcopts

This is a list of fields in `-lavcopts` that you may be required to change in order to make a complaint movie for VCD, SVCD, or DVD:

- **acodec**: `mp2` for VCD, SVCD, or PAL DVD; `ac3` is most commonly used for DVD. PCM audio may also be used for DVD, but this is mostly a big waste of space. Note that MP3 audio is not compliant for any of these formats, but players often have no problem playing it anyway.
- **abitrate**: 224 for VCD; up to 384 for SVCD; up to 1536 for DVD, but commonly used values range from 192 kbps for stereo to 384 kbps for 5.1 channel sound.
- **vcodec**: `mpeg1video` for VCD; `mpeg2video` for SVCD; `mpeg2video` is usually used for DVD but you may also use `mpeg1video` for CIF resolutions.
- **keyint**: Used to set the GOP size. 18 for 30fps material, or 15 for 25/24 fps material. Commercial producers seem to prefer keyframe intervals of 12. It is possible to make this much larger and still retain compatibility with most players. A `keyint` of 25 should never cause any problems.
- **vrc_buf_size**: 327 for VCD, 917 for SVCD, and 1835 for DVD.
- **vrc_minrate**: 1152, for VCD. May be left alone for SVCD and DVD.
- **vrc_maxrate**: 1152 for VCD; 2500 for SVCD; 9800 for DVD. For SVCD and DVD, you might wish to use lower values depending on your own personal preferences and requirements.
- **vbitrate**: 1152 for VCD; up to 2500 for SVCD; up to 9800 for DVD. For the latter two formats, vbitrate should be set based on personal preference. For instance, if you insist on fitting 20 or so hours on a DVD, you could use vbitrate=400. The resulting video quality would probably be quite bad. If you are trying to squeeze out the maximum possible quality on a DVD, use vbitrate=9800, but be warned that this could constrain you to less than an hour of video on a single-layer DVD.
- **vstrict**: `vstrict`=0 should be used to create DVDs. Without this option, MEncoder creates a stream that cannot be correctly decoded by some standalone DVD players.

11.8.3.3. Examples

This is a typical minimum set of `-lavcopts` for encoding video:

VCD:

```
-lavcopts vcodec=mpeg1video:vrc_buf_size=327:vrc_minrate=1152:\
vrc_maxrate=1152:vbitrate=1152:keyint=15:acodec=mp2
```

SVCD:

```
-lavcopts vcodec=mpeg2video:vrc_buf_size=917:vrc_maxrate=2500:vbitrate=1800:\
keyint=15:acodec=mp2
```

DVD:

```
-lavcopts vcodec=mpeg2video:vrc_buf_size=1835:vrc_maxrate=9800:vbitrate=5000:\
keyint=15:vstrict=0:acodec=ac3
```

11.8.3.4. Advanced Options

For higher quality encoding, you may also wish to add quality-enhancing options to lavcopts, such as `trell`, `mbd=2`, and others. Note that `qpel` and `v4mv`, while often useful with MPEG-4, are not usable with MPEG-1 or MPEG-2. Also, if you are trying to make a very high quality DVD encode, it may be useful to add `dc=10` to lavcopts. Doing so may help reduce the appearance of blocks in flat-colored areas. Putting it all together, this is an example of a set of lavcopts for a higher quality DVD:

```
-lavcopts vcodec=mpeg2video:vrc_buf_size=1835:vrc_maxrate=9800:vbitrate=8000:\
keyint=15:trell:mbd=2:precmp=2:subcmp=2:cmp=2:dia=-10:predia=-10:cbp:mv0:\
vqmin=1:lmin=1:dc=10:vstrict=0
```

11.8.4. Encoding Audio

VCD and SVCD support MPEG-1 layer II audio, using one of `toolame`, `twolame`, or `libavcodec`'s MP2 encoder. The libavcodec MP2 is far from being as good as the other two libraries, however it should always be available to use. VCD only supports constant bitrate audio (CBR) whereas SVCD supports variable bitrate (VBR), too. Be careful when using VBR because some bad standalone players might not support it too well.

For DVD audio, `libavcodec`'s AC-3 codec is used.

11.8.4.1. toolame

For VCD and SVCD:

```
-oac toolame -toolameopts br=224
```

11.8.4.2. twolame

For VCD and SVCD:

```
-oac twolame -twolameopts br=224
```

11.8.4.3. libavcodec

For DVD with 2 channel sound:

```
-oac lavc -lavcopts acodec=ac3:abitrate=192
```

For DVD with 5.1 channel sound:

```
-channels 6 -oac lavc -lavcopts acodec=ac3:abitrate=384
```

For VCD and SVCD:

```
-oac lavc -lavcopts acodec=mp2:abitrate=224
```

11.8.5. Putting it all Together

This section shows some complete commands for creating VCD/SVCD/DVD compliant videos.

11.8.5.1. PAL DVD

```
mencoder -oac lavc -ovc lavc -of mpeg -mpegopts format=dvd:tsaf \
  -vf scale=720:576,harddup -srate 48000 -af lavcresample=48000 \
  -lavcopts vcodec=mpeg2video:vrc_buf_size=1835:vrc_maxrate=9800:vbitrate=5000:\
keyint=15:vstrict=0:acodec=ac3:abitrate=192:aspect=16/9 -ofps 25 \
  -o movie.mpg movie.avi
```

11.8.5.2. NTSC DVD

```
mencoder -oac lavc -ovc lavc -of mpeg -mpegopts format=dvd:tsaf \
  -vf scale=720:480,harddup -srate 48000 -af lavcresample=48000 \
  -lavcopts vcodec=mpeg2video:vrc_buf_size=1835:vrc_maxrate=9800:vbitrate=5000:\
keyint=18:vstrict=0:acodec=ac3:abitrate=192:aspect=16/9 -ofps 30000/1001 \
  -o movie.mpg movie.avi
```

11.8.5.3. PAL AVI Containing AC-3 Audio to DVD

If the source already has AC-3 audio, use -oac copy instead of re-encoding it.

```
mencoder -oac copy -ovc lavc -of mpeg -mpegopts format=dvd:tsaf \
  -vf scale=720:576,harddup -ofps 25 \
  -lavcopts vcodec=mpeg2video:vrc_buf_size=1835:vrc_maxrate=9800:vbitrate=5000:\
keyint=15:vstrict=0:aspect=16/9 -o movie.mpg movie.avi
```

11.8.5.4. NTSC AVI Containing AC-3 Audio to DVD

If the source already has AC-3 audio, and is NTSC @ 24000/1001 fps:

```
mencoder -oac copy -ovc lavc -of mpeg -mpegopts format=dvd:tsaf:telecine \
  -vf scale=720:480,harddup -lavcopts vcodec=mpeg2video:vrc_buf_size=1835:\
  vrc_maxrate=9800:vbitrate=5000:keyint=15:vstrict=0:aspect=16/9 -ofps 24000/1001 \
  -o movie.mpg movie.avi
```

11.8.5.5. PAL SVCD

```
mencoder -oac lavc -ovc lavc -of mpeg -mpegopts format=xsvcd -vf \
    scale=480:576,harddup -srate 44100 -af lavcresample=44100 -lavcopts \
    vcodec=mpeg2video:mbd=2:keyint=15:vrc_buf_size=917:vrc_minrate=600:\
vbitrate=2500:vrc_maxrate=2500:acodec=mp2:abitrate=224:aspect=16/9 -ofps 25 \
    -o movie.mpg movie.avi
```

11.8.5.6. NTSC SVCD

```
mencoder -oac lavc -ovc lavc -of mpeg -mpegopts format=xsvcd  -vf \
    scale=480:480,harddup -srate 44100 -af lavcresample=44100 -lavcopts \
    vcodec=mpeg2video:mbd=2:keyint=18:vrc_buf_size=917:vrc_minrate=600:\
vbitrate=2500:vrc_maxrate=2500:acodec=mp2:abitrate=224:aspect=16/9 -ofps 30000/1001 \
    -o movie.mpg movie.avi
```

11.8.5.7. PAL VCD

```
mencoder -oac lavc -ovc lavc -of mpeg -mpegopts format=xvcd -vf \
    scale=352:288,harddup -srate 44100 -af lavcresample=44100 -lavcopts \
    vcodec=mpeg1video:keyint=15:vrc_buf_size=327:vrc_minrate=1152:\
vbitrate=1152:vrc_maxrate=1152:acodec=mp2:abitrate=224:aspect=16/9 -ofps 25 \
    -o movie.mpg movie.avi
```

11.8.5.8. NTSC VCD

```
mencoder -oac lavc -ovc lavc -of mpeg -mpegopts format=xvcd -vf \
    scale=352:240,harddup -srate 44100 -af lavcresample=44100 -lavcopts \
    vcodec=mpeg1video:keyint=18:vrc_buf_size=327:vrc_minrate=1152:\
vbitrate=1152:vrc_maxrate=1152:acodec=mp2:abitrate=224:aspect=16/9 -ofps 30000/1001 \
    -o movie.mpg movie.avi
```

[1] Be careful, however: Decoding DVD-resolution MPEG-4 AVC videos requires a fast machine (i.e. a Pentium 4 over 1.5GHz or a Pentium M over 1GHz).

[2] The same encode may not look the same on someone else's monitor or when played back by a different decoder, so future-proof your encodes by playing them back on different setups.

Chapter 12. Frequently Asked Questions

12.1. Development

Q: How do I create a proper patch for MPlayer?

A: We made a short document describing all the necessary details. Please follow the instructions.

Q: How do I translate MPlayer to a new language?

A: Read the translation HOWTO, it should explain everything. You can get further help on the MPlayer-translations mailing

Q: How can I support MPlayer development?

A: We are more than happy to accept your hardware and software donations. They help us in continuously improving MPla

Q: How can I become an MPlayer developer?

A: We always welcome coders and documenters. Read the technical documentation to get a first grasp. Then you should s dev-eng mailing list and start coding. If you want to help out with the documentation, join the MPlayer-docs mailing list.

Q: Why don't you use autoconf/automake?

A: We have a modular, handwritten build system. It does a reasonably good job, so why change? Besides, we dislike the a people.

12.2. Compilation and installation

Q: Compilation fails with an error and gcc bails out with some cryptic message containing the phrase `internal compile` `find a register to spill` or `can't find a register in class `GENERAL_REGS' while reloading`

A: You have stumbled over a bug in gcc. Please report it to the gcc team but not to us. For some reason MPlayer seems to frequently. Nevertheless we cannot fix them and do not add workarounds for compiler bugs to our sources. To avoid this compiler version that is known to be reliable and stable, or upgrade frequently.

Q: Are there binary (RPM/Debian) packages of MPlayer?

A: See the Debian and RPM section for details.

Q: How can I build a 32 bit MPlayer on a 64 bit Athlon?

A: Try the following configure options:

```
./configure --target=i386-linux --cc="gcc -m32" --as="as --32" --with-extralibdir=/usr/l
```

Q: Configure ends with this text, and MPlayer won't compile!

```
Your gcc does not support even i386 for '-march' and '-mcpu'
```

A: Your gcc isn't installed correctly, check the `configure.log` file for details.

Q: I have a Matrox G200/G400/G450/G550, how do I compile/use the `mga_vid` driver?

A: Read the mga_vid section.

Q: During 'make', MPlayer complains about missing X11 libraries. I don't understand, I *do* have X11 installed!?

A: ... but you don't have the X11 development package installed. Or not correctly. It's called `XFree86-devel*` under Red Debian Woody and `libx11-dev` under Debian Sarge. Also check if the `/usr/X11` and `/usr/include/X11` symlinks

Q: Building on Mac OS 10.3 leads to several link errors.

A: The link error you're experiencing most likely looks like this:

```
ld: Undefined symbols:
_LLCStyleInfoCheckForOpenTypeTables referenced from QuartzCore expected to be defined in
_LLCStyleInfoGetUserRunFeatures referenced from QuartzCore expected to be defined in App
```

This problem is the result of Apple developers using 10.4 to compile their software and distributing the binaries to 10.3 u
The undefined symbols are present in Mac OS 10.4, but not 10.3. One solution can be to downgrade to QuickTime 7.0.1

Get an older copy of the frameworks. This will give you a compressed file that contains the QuickTime 7.0.1 Framework
Framework.

Uncompress the files somewhere that is not in your System folder. (i.e. do not install these frameworks into your `/System/Library/Frameworks`! Using this older copy is only meant to get around link errors!)

```
gunzip < CompatFrameworks.tgz | tar xvf -
```

In config.mak, you should append `-F/path/to/where/you/extracted` to the `OPTFLAGS` variable. If you use X-Cod
frameworks instead of the system ones.

The resulting MPlayer binary will actually use the framework that is installed on your system via dynamic links that are re
can verify this using `otool -l`).

12.3. General questions

Q: Are there any mailing lists on MPlayer?

A: Yes. Look at the mailing lists section of our homepage.

Q: I've found a nasty bug when I tried to play my favorite video! Who should I inform?

A: Please read the bug reporting guidelines and follow the instructions.

Q: I get a core dump when trying to dump streams, what's wrong?

A: Don't panic. Make sure you know where your towel is.

Seriously, notice the smiley and start looking for files that end in `.dump`.

Q: When I start playing, I get this message but everything seems fine:

```
Linux RTC init: ioctl (rtc_pie_on): Permission denied
```

A: You need a specially set up kernel to use the RTC timing code. For details see the RTC section of the documentation.

Q: How can I make a screenshot?

A: You have to use a video output driver that does not employ an overlay to be able to take a screenshot. Under X11, `-vo`
Windows `-vo directx:noaccel` works.

Alternatively you can run MPlayer with the `screenshot` video filter (`-vf screenshot`), and press the **s** key to grab a

Q: What is the meaning of the numbers on the status line?

A: Example:

```
A: 2.1 V: 2.2 A-V: -0.167 ct: 0.042 57/57 41% 0% 2.6% 0 4 49% 1.00x
```

`A: 2.1`

audio position in seconds

`V: 2.2`

video position in seconds

`A-V: -0.167`

audio-video difference in seconds (delay)

`ct: 0.042`

total A-V sync correction done

`57/57`

frames played/decoded (counting from last seek)

`41%`

video codec CPU usage in percent (for slice rendering and direct rendering this includes video_out)

`0%`

video_out CPU usage

`2.6%`

audio codec CPU usage in percent

`0`

frames dropped to maintain A-V sync

`4`

current level of image postprocessing (when using `-autoq`)

`49%`

current cache size used (around 50% is normal)

`1.00x`

playback speed as a factor of original speed

Most of them are for debug purposes, use the `-quiet` option to make them disappear. You might notice that video_out
some files. This is because it is called directly from the codec and thus cannot be measured separately. If you wish to kn
compare the difference when playing the file with `-vo null` and your usual video output driver.

Q: There are error messages about file not found `/usr/local/lib/codecs/` ...

A: Download and install the binary codecs from our download page.

Q: How can I make MPlayer remember the options I use for a particular file, e.g. `movie.avi`?

A: Create a file named `movie.avi.conf` with the file-specific options in it and put it in `~/.mplayer` or in the same directo

Q: Subtitles are very nice, the most beautiful I've ever seen, but they slow down playing! I know it's unlikely ...

A: After running `./configure`, edit `config.h` and replace `#undef FAST_OSD` with `#define FAST_OSD`. Then recomp

Q: How can I run MPlayer in the background?

A: Use:

```
mplayer options filename < /dev/null &
```

12.4. Playback problems

Q: I cannot pinpoint the cause of some strange playback problem.

A: Do you have a stray `codecs.conf` file in `~/.mplayer/`, `/etc/`, `/usr/local/etc/` or a similar location? Remove it,
file can cause obscure problems and is intended for use only by developers working on codec support. It overrides MPla
settings, which will wreak havoc if incompatible changes are made in newer program versions. Unless used by experts i
the form of seemingly random and very hard to localize crashes and playback problems. If you still have it somewhere o
remove it now.

Q: How can I get subtitles to appear on the black margins around a movie?

A: Use the `expand` video filter to increase the area onto which the movie is rendered vertically and place the movie at the t

```
mplayer -vf expand=0:-100:0:0 -slang de dvd://1
```

Q: How can I select audio/subtitle tracks from a DVD, OGM, Matroska or NUT file?

A: You have to use `-aid` (audio ID) or `-alang` (audio language), `-sid`(subtitle ID) or `-slang` (subtitle language), for exa

```
mplayer -alang eng -slang eng example.mkv
mplayer -aid 1 -sid 1 example.mkv
```

To see which ones are available:

```
mplayer -vo null -ao null -frames 0 -v filename | grep sid
mplayer -vo null -ao null -frames 0 -v filename | grep aid
```

Q: I'm trying to play a random stream off the internet but it fails.

A: Try playing the stream with the `-playlist` option.

Q: I downloaded a movie off a P2P network and it doesn't work!

A: Your file is most probably broken or a fake file. If you got it from a friend, and he says it works, try comparing md5sum h

Q: I'm having trouble getting my subtitles to display, help!!

A: Make sure you have installed fonts properly. Run through the steps in the Fonts and OSD part of the installation section TrueType fonts, verify that you have the `FreeType` library installed. Other things include checking your subtitles in a tex players. Also try converting them to another format.

Q: Why doesn't MPlayer work on Fedora Core?

A: There is a bad interaction on Fedora between exec-shield, prelink, and any applications which use Windows DLLs (such

The problem is that exec-shield randomizes the load addresses of all the system libraries. This randomization happens two weeks).

When MPlayer tries to load a Windows DLL it wants to put it at a specific address (0x400000). If an important system lib already, MPlayer will crash. (A typical symptom would be a segmentation fault when trying to play Windows Media 9 file

If you run into this problem you have two options:

- Wait two weeks. It might start working again.
- Relink all the binaries on the system with different prelink options. Here are step by step instructions:
  1. Edit `/etc/syconfig/prelink` and change

     ```
     PRELINK_OPTS=-mR
     ```

     to

     ```
     PRELINK_OPTS="-mR --no-exec-shield"
     ```

  2. **touch /var/lib/misc/prelink.force**
  3. **/etc/cron.daily/prelink** (This relinks all the applications, and it takes quite a while.)
  4. **execstack -s *`/path/to/`*mplayer** (This turns off exec-shield for the MPlayer binary.)

Q: MPlayer dies with

```
MPlayer interrupted by signal 4 in module: decode_video
```

A: Don't use MPlayer on a CPU different from the one it was compiled on or recompile with runtime CPU detection (**./confi cpudetection**).

Q: When I try to grab from my tuner, it works, but colors are strange. It's OK with other applications.

A: Your card probably reports some colorspaces as supported when in fact it does not support them. Try with YUY2 instea the TV section).

Q: I get very strange percentage values (way too big) while playing files on my notebook.

A: It's an effect of the power management / power saving system of your notebook (BIOS, not kernel). Plug the external po you power on your notebook. You can also try whether cpufreq (a SpeedStep interface for Linux) helps you.

Q: The audio/video gets totally out of sync when I run MPlayer as `root` on my notebook. It works normal when i run it as a

A: This is again a power management effect (see above). Plug the external power connector in **before** you power on your do not use the `-rtc` option.

Q: While playing a movie it suddenly gets jerky and I get the following message:

```
Badly interleaved AVI file detected - switching to -ni mode...
```

A: Badly interleaved files and `-cache` don’t work well together. Try `-nocache`.

12.5. Video/audio driver problems (vo/ao)

Q: When I go into fullscreen mode I just get black borders around the image and no real scaling to fullscreen mode.

A: Your video output driver does not support scaling in hardware and since scaling in software can be incredibly slow MPla
enable it. Most likely you are using the `x11` instead of the `xv` video output driver. Try adding `-vo xv` to the command li
to find out about alternative video output drivers. The `-zoom` option explicitly enables software scaling.

Q: I’ve just installed MPlayer. When I want to open a video file it causes a fatal error:

```
Error opening/initializing the selected video_out (-vo) device.
```

How can I solve my problem?

A: Just change your video output device. Issue the following command to get a list of available video output drivers:

```
mplayer -vo help
```

After you’ve chosen the correct video output driver, add it to your configuration file. Add

```
vo = selected_vo
```

to `~/.mplayer/config` and/or

```
vo_driver = selected_vo
```

to `~/.mplayer/gui.conf`.

Q: I have problems with *`[your window manager]`* and fullscreen xv/xmga/sdl/x11 modes ...

A: Read the bug reporting guidelines and send us a proper bug report. Also try experimenting with the `-fstype` option.

Q: Audio goes out of sync playing an AVI file.

A: Try the `-bps` or `-nobps` option. If it does not improve, read the bug reporting guidelines and upload the file to FTP.

Q: How can I use dmix with MPlayer?

A: After setting up your asoundrc you have to use `-ao alsa:device=dmix`.

Q: I have no sound when playing a video and get error messages similar to this one:

```
AO: [oss] 44100Hz 2ch Signed 16-bit (Little-Endian)
[AO OSS] audio_setup: Can't open audio device /dev/dsp: Device or resource busy
Could not open/initialize audio device -> no sound.
Audio: no sound
Starting playback...
```

A: Are you running KDE or GNOME with the aRts or ESD sound daemon? Try disabling the sound daemon or use the `-ao`
to make MPlayer use aRts or ESD. You might also be running ALSA without OSS emulation, try loading the ALSA OSS
`-ao alsa` to your command line to directly use the ALSA audio output driver.

Q: When starting MPlayer under KDE I just get a black screen and nothing happens. After about one minute the video start

A: The KDE aRts sound daemon is blocking the sound device. Either wait until the video starts or disable the aRts daemon
want to use aRts sound, specify audio output via our native aRts audio driver (`-ao arts`). If it fails or isn't compiled in,
make sure your SDL can handle aRts sound. Yet another option is to start MPlayer with artsdsp.

Q: I have A/V sync problems. Some of my AVIs play fine, but some play with double speed!

A: You have a buggy sound card/driver. Most likely it's fixed at 44100Hz, and you try to play a file which has 22050Hz audi
filter.

Q: When I play this movie I get video-audio desync and/or MPlayer crashes with the following message:

```
Too many (945 in 8390980 bytes) video packets in the buffer!
```

A: This can have multiple reasons.

- Your CPU *and/or* video card *and/or* bus is too slow. MPlayer displays a message if this is the case (and the dropp
fast).
- If it is an AVI, maybe it has bad interleaving, try the `-ni` option to work around this. Or it may have a bad header,
`-mc 0` can help.
- Many FLV files will only play correctly with `-correct-pts`. Unfortunately MEncoder does not have this option, b
to the correct value manually if you know it.
- Your sound driver is buggy.

Q: How can I get rid of A/V desynchronization while seeking through RealMedia streams?

A: `-mc 0.1` can help.

12.6. DVD playback

Q: What about DVD navigation/menus?

A: MPlayer does not support DVD menus due to serious architectural limitations that prevent proper handling of still images
you want to have fancy menus, you will have to use another player like xine, VLC or Ogle. If you want to see DVD navig
have to implement it yourself, but be aware that it is a major undertaking.

Q: What about subtitles? Can MPlayer display them?

A: Yes. See the DVD chapter.

Q: How can I set the region code of my DVD-drive? I don't have Windows!

A: Use the regionset tool.

Q: I can't play a DVD, MPlayer hangs or outputs "Encrypted VOB file!" errors.

A: CSS decryption code does not work with some DVD drives unless you set the region code appropriately. See the answe

Q: Do I need to be (setuid) root to be able to play a DVD?

A: No. However you must have the proper rights on the DVD device entry (in `/dev/`).

Q: Is it possible to play/encode only selected chapters?

A: Yes, try the `-chapter` option.

Q: My DVD playback is sluggish!

A: Use the `-cache` option (described in the man page) and try enabling DMA for the DVD drive with the **hdparm** tool (des

Q: I copied a DVD using vobcopy. How do I play/encode it from my hard disk?

A: Use the `-dvd-device` option to refer to the directory that contains the files:

```
mplayer dvd://1 -dvd-device /path/to/directory
```

12.7. Feature requests

Q: If MPlayer is paused and I try to seek or press any key at all, MPlayer ceases to be paused. I would like to be able to se

A: This is very tricky to implement without losing A/V synchronization. All attempts have failed so far, but patches are welco

Q: I'd like to seek +/- 1 frame instead of 10 seconds.

A: You can step forward one frame by pressing **.**. If the movie was not paused it will be paused afterwards (see the man pa backwards is unlikely to be implemented anytime soon.

12.8. Encoding

Q: How can I encode?

A: Read the MEncoder section.

Q: How can I dump a full DVD title into a file?

A: Once you have selected your title, and made sure it plays fine with MPlayer, use the option `-dumpstream`. For example

```
mplayer dvd://5 -dumpstream -dumpfile dvd_dump.vob
```

will dump the 5th title of the DVD into the file *dvd_dump.vob*

Q: How can I create (S)VCDs automatically?

A: Try the `mencvcd.sh` script from the `TOOLS` subdirectory. With it you can encode DVDs or other movies to VCD or SVCI them directly to CD.

Q: How can I create (S)VCDs?

A: Newer versions of MEncoder can directly generate MPEG-2 files that can be used as a base to create a VCD or SVCD a out of the box on all platforms (for example, to share a video from a digital camcorder with your computer-illiterate friend MEncoder to create VCD/SVCD/DVD-compliant files for more details.

Q: How can I join two video files?

A: MPEG files can be concatenated into a single file with luck. For AVI files, you can use MEncoder's multiple file support li

```
mencoder -ovc copy -oac copy -o out.avi file1.avi file2.avi
```

This will only work if the files are of the same resolution and use the same codec. You can also try avidemux and avimer tool set).

Q: How can I fix AVI files with a broken index or bad interleaving?

A: To avoid having to use `-idx` to be able to seek in AVI files with a broken index or `-ni` to play AVI files with bad interlea

```
mencoder input.avi -idx -ovc copy -oac copy -o output.avi
```

to copy the video and audio streams into a new AVI file while regenerating the index and correctly interleaving the data. possible bugs in the video and/or audio streams.

Q: How can I fix the aspect ratio of an AVI file?

A: You can do such a thing thanks to MEncoder's `-force-avi-aspect` option, which overrides the aspect stored in the A option. For example:

```
mencoder input.avi -ovc copy -oac copy -o output.avi -force-avi-aspect 4/3
```

Q: How can I backup and encode a VOB file with a broken beginning?

A: The main problem when you want to encode a VOB file which is corrupted [3] is that it will be hard to get an encode with workaround is to just shave off the corrupted part and encode just the clean part. First you need to find where the clean

```
mplayer input.vob -sb nb_of_bytes_to_skip
```

Then you can create a new file which contains just the clean part:

```
dd if=input.vob of=output_cut.vob skip=1 ibs=nb_of_bytes_to_skip
```

Q: I can't encode DVD subtitles into the AVI!

A: You have to properly specify the `-sid` option.

Q: How can I encode only selected chapters from a DVD?

A: Use the `-chapter` option correctly, like: `-chapter 5-7`.

Q: I'm trying to work with 2GB+ files on a VFAT file system. Does it work?

A: No, VFAT doesn't support 2GB+ files.

Q: What is the meaning of the numbers on the status line during the encoding process?

A: Example:

```
Pos: 264.5s   6612f ( 2%)  7.12fps Trem: 576min 2856mb  A-V:0.065 [2126:192]
```

`Pos: 264.5s`

time position in the encoded stream

`6612f`

number of video frames encoded

`( 2%)`

portion of the input stream encoded

`7.12fps`

encoding speed

`Trem: 576min`

estimated remaining encoding time

`2856mb`

estimated size of the final encode

`A-V:0.065`

current delay between audio and video streams

`[2126:192]`

average video bitrate (in kb/s) and average audio bitrate (in kb/s)

Q: Why is the recommended bitrate printed by MEncoder negative?

A: Because the bitrate you encoded the audio with is too large to fit the movie on any CD. Check if you have libmp3lame in

Q: I can't encode an ASF file to AVI/MPEG-4 (DivX) because it uses 1000 fps.

A: Since ASF uses variable framerate but AVI uses a fixed one, you have to set it by hand with the `-ofps` option.

Q: How can I put subtitles in the output file?

A: Just pass the `-sub <filename>` (or `-sid`, respectively) option to MEncoder.

Q: How do I encode only sound from a music video?

A: It's not possible directly, but you can try this (note the **&** at the end of **mplayer** command):

```
mkfifo encode
mplayer -ao pcm -aofile encode dvd://1 &
lame your_opts encode music.mp3
rm encode
```

This allows you to use any encoder, not only LAME, just replace **lame** with your favorite audio encoder in the above con

Q: Why do third-party players fail to play MPEG-4 movies encoded by MEncoder versions later than 1.0pre7?

A: `libavcodec`, the native MPEG-4 encoding library usually shipped with MEncoder, used to set the FourCC to 'DIVX' wh videos (the FourCC is an AVI tag to identify the software used to encode and the intended software to use for decoding people to think that `libavcodec` was a DivX encoding library, when in fact it is a completely different MPEG-4 encodin the MPEG-4 standard much better than DivX does. Therefore, the new default FourCC used by `libavcodec` is 'FMP4', behavior using MEncoder's `-ffourcc` option. You may also change the FourCC of existing files in the same way:

```
mencoder input.avi -o output.avi -ffourcc XVID
```

Note that this will set the FourCC to XVID rather than DIVX. This is recommended as DIVX FourCC means DivX4, whicl codec, whereas DX50 and XVID both mean full MPEG-4 (ASP). Therefore, if you change the FourCC to DIVX, some ba players may choke on some advanced features that `libavcodec` supports, but DivX doesn't; on the other hand `Xvid` is terms of functionality, and is supported by all decent players.

Q: How can I encode an audio only file?

A: Use `aconvert.sh` from the `TOOLS` subdirectory in the MPlayer source tree.

Q: How can I play subtitles embedded in AVI?

A: Use `avisubdump.c` from the `TOOLS` subdirectory or read this document about extracting/demultiplexing subtitles embe

Q: MEncoder won't...

A: Have a look at the `TOOLS` subdirectory for a collection of random scripts and hacks. `TOOLS/README` contains document

---

[3] To some extent, some forms of copy protection used in DVDs can be assumed to be content corruption.

Appendix A. How to report bugs

Good bug reports are a very valuable contribution to the development of any software project. But just like writing good software, good problem reports involve some work. Please realize that most developers are extremely busy and receive obscene amounts of email. So while your feedback is crucial in improving MPlayer and very much appreciated, please understand that you have to provide **all** of the information we request and follow the instructions in this document closely.

A.1. Report security related bugs

In case you have found an exploitable bug and you would like to do the right thing and let us fix it before you disclose it, we would be happy to get your security advisory at security@mplayerhq.hu. Please add [SECURITY] or [ADVISORY] in the subject. Be sure that your report contains complete and detailed analysis of the bug. Sending a fix is highly appreciated. Please don't delay your report to write proof-of-concept exploit, you can send that one with another mail.

A.2. How to fix bugs

If you feel have the necessary skills you are invited to have a go at fixing the bug yourself. Or maybe you already did that? Please read this short document to find out how to get your code included in MPlayer. The people on the MPlayer-dev-eng mailing list will assist you if you have questions.

A.3. How to do regression testing using Subversion

A problem that can happen sometimes is 'it used to work before, now it doesn't anymore...'. Here is a step by step procedure to try to pinpoint when the problem occurred. This is **not** for casual users.

First, you'd need to fetch MPlayer's source tree from Subversion. Instructions can be found in the Subversion section of the download page.

You will have now in the mplayer/ directory an image of the Subversion tree, on the client side. Now update this image to the date you want:

```
cd mplayer/
svn update -r {"2004-08-23"}
```

The date format is YYYY-MM-DD HH:MM:SS. Using this date format ensure that you will be able to extract patches according to the date at which they were committed, as in the MPlayer-cvslog archive.

Now proceed as for a normal update:

```
./configure
make
```

If any non-programmer reads this, the fastest method to get at the point where the problem occurred is to use a binary search — that is, search the date of the breakage by repeatedly dividing the search interval in half. For example, if the problem occurred in 2003, start at mid-year, then ask "Is the problem already here?". If yes, go back to the first of April; if not, go to the first of October, and so on.

If you have lot of free hard disk space (a full compile currently takes 100 MB, and around 300-350 MB if debugging symbols are enabled), copy the oldest known working version before updating it; this will save time if you need to go back. (It is usually necessary to run 'make distclean' before recompiling an earlier version, so if you do not make a backup copy of your original source tree, you will have to recompile everything in it when you come back to the present.) Alternatively you may use ccache to speed up compilation.

When you have found the day where the problem happened, continue the search using the mplayer-cvslog archive (sorted by date) and a more precise svn update including hour, minute and second:

```
svn update -r {"2004-08-23 15:17:25"}
```

This will allow you to easily find the exact patch that did it.

If you find the patch that is the cause of the problem, you have almost won; report about it to the MPlayer Bugzilla or subscribe to MPlayer-users and post it there. There is a chance that the author will jump in to suggest a fix. You may also look hard at the patch until it is coerced to reveal where the bug is :-).

A.4. How to report bugs

First of all please try the latest Subversion version of MPlayer as your bug might already be fixed there. Development moves extremely fast, most problems in official releases are reported within days or even hours, so please use **only Subversion** to report bugs. This includes binary packages of MPlayer. Subversion instructions can be found at the bottom of this page or in the README. If this did not help please refer to the rest of the documentation. If your problem is not known or not solvable by our instructions, then please report the bug.

Please do not send bug reports privately to individual developers. This is community work and thus there might be several people interested in it. Sometimes other users already experienced your troubles and know how to circumvent a problem even if it is a bug in MPlayer code.

Please describe your problem in as much detail as possible. Do a little detective work to narrow down the circumstances under which the problem occurs. Does the bug only show up in certain situations? Is it specific to certain files or file types? Does it occur with only one codec or is it codec independent? Can you reproduce it with all output drivers? The more information you provide the better are our chances at fixing your problem. Please do not forget to also include the valuable information requested below, we will be unable to properly diagnose your problem otherwise.

An excellent and well written guide to asking questions in public forums is How To Ask Questions The Smart Way by Eric S. Raymond. There is another called How to Report Bugs Effectively by Simon Tatham. If you follow these guidelines you should be able to get help. But please understand that we all follow the mailing lists voluntarily in our free time. We are very busy and cannot guarantee that you will get a solution for your problem or even an answer.

A.5. Where to report bugs

Subscribe to the MPlayer-users mailing list: http://lists.mplayerhq.hu/mailman/listinfo/mplayer-users and send your bug report to mailto:mplayer-users@mplayerhq.hu where you can discuss it.

If you prefer, you can use our brand-new Bugzilla instead.

The language of this list is **English**. Please follow the standard Netiquette Guidelines and **do not send HTML mail** to any of our mailing lists. You will only get ignored or banned. If you do not know what HTML mail is or why it is evil, read this fine document. It explains all the details and has instructions for turning HTML off. Also note that we will not individually CC (carbon-copy) people so it is a good idea to subscribe to actually receive your answer.

A.6. What to report

You may need to include log, configuration or sample files in your bug report. If some of them are quite big then it is better to upload them to our FTP server in a compressed format (gzip and bzip2 preferred) and include only the path and file name in your bug report. Our mailing lists have a message size limit of 80k, if you have something bigger you have to compress or upload it.

A.6.1. System Information

- Your Linux distribution or operating system and version e.g.:
  - Red Hat 7.1
  - Slackware 7.0 + devel packs from 7.1 ...
- kernel version:

  ```
  uname -a
  ```
- libc version:

  ```
  ls -l /lib/libc[.-]*
  ```

- gcc and ld versions:

  ```
  gcc -v
  ld -v
  ```

- binutils version:

  ```
  as --version
  ```

- If you have problems with fullscreen mode:
  - Window manager type and version
- If you have problems with XVIDIX:
  - X colour depth:

    ```
    xdpyinfo | grep "depth of root"
    ```

- If only the GUI is buggy:
  - GTK version
  - GLIB version
  - GUI situation in which the bug occurs

A.6.2. Hardware and drivers

- CPU info (this works on Linux only):

  ```
  cat /proc/cpuinfo
  ```

- Video card manufacturer and model, e.g.:
  - ASUS V3800U chip: nVidia TNT2 Ultra pro 32MB SDRAM
  - Matrox G400 DH 32MB SGRAM
- Video driver type & version, e.g.:
  - X built-in driver
  - nVidia 0.9.623
  - Utah-GLX CVS 2001-02-17
  - DRI from X 4.0.3
- Sound card type & driver, e.g.:
  - Creative SBLive! Gold with OSS driver from oss.creative.com
  - Creative SB16 with kernel OSS drivers
  - GUS PnP with ALSA OSS emulation
- If in doubt include **lspci -vv** output on Linux systems.

A.6.3. Configure problems

If you get errors while running **./configure**, or autodetection of something fails, read `configure.log`. You may find the answer there, for example multiple versions of the same library mixed on your system, or you forgot to install the development package (those with the -dev suffix). If you think there is a bug, include `configure.log` in your bug report.

A.6.4. Compilation problems

Please include these files:

- config.h
- config.mak

A.6.5. Playback problems

Please include the output of MPlayer at verbosity level 1, but remember to **not truncate the output** when you paste it into your mail. The developers need all of the messages to properly diagnose a problem. You can direct the output into a file like this:

```
mplayer -v options filename > mplayer.log 2>&1
```

If your problem is specific to one or more files, then please upload the offender(s) to: ftp://upload.mplayerhq.hu/MPlayer/incoming/

Also upload a small text file having the same base name as your file with a .txt extension. Describe the problem you are having with the particular file there and include your email address as well as the output of MPlayer at verbosity level 1. Usually the first 1-5 MB of a file are enough to reproduce the problem, but to be sure we ask you to:

```
dd if=yourfile of=smallfile bs=1024k count=5
```

It will take the first five megabytes of '**your-file**' and write it to '**small-file**'. Then try again on this small file and if the bug still shows up your sample is sufficient for us. Please **do not ever** send such files via mail! Upload it, and send only the path/filename of the file on the FTP-server. If the file is accessible on the net, then sending the **exact** URL is sufficient.

A.6.6. Crashes

You have to run MPlayer inside **gdb** and send us the complete output or if you have a `core` dump of the crash you can extract useful information from the Core file. Here's how:

A.6.6.1. How to conserve information about a reproducible crash

Recompile MPlayer with debugging code enabled:

```
./configure --enable-debug=3
make
```

and then run MPlayer within gdb using:

```
gdb ./mplayer
```

You are now within gdb. Type:

```
run -v options-to-mplayer filename
```

and reproduce your crash. As soon as you did it, gdb will return you to the command line prompt where you have to enter

```
bt
disass $pc-32 $pc+32
info all-registers
```

A.6.6.2. How to extract meaningful information from a core dump

Create the following command file:

```
bt
disass $pc-32 $pc+32
info all-registers
```

Then simply execute this command:

```
gdb mplayer --core=core -batch --command=command_file > mplayer.bug
```

A.7. I know what I am doing...

If you created a proper bug report following the steps above and you are confident it is a bug in MPlayer, not a compiler problem or broken file, you have already read the documentation and you could not find a solution, your sound drivers are OK, then you might want to subscribe to the MPlayer-advusers list and send your bug report there to get a better and faster answer.

Please be advised that if you post newbie questions or questions answered in the manual there, you will be ignored or flamed instead of getting an appropriate answer. So do not flame us and subscribe to -advusers only if you really know what you are doing and feel like being an advanced MPlayer user or developer. If you meet these criteria it should not be difficult to find out how to subscribe...

Appendix B. MPlayer skin format

B.1. Overview

B.1.1. Skin components

Skins are quite free-format (unlike the fixed-format skins of Winamp/XMMS, for example), so it is up to you to create something great.

Currently there are four windows to be decorated: the main window, the subwindow, the playbar, and the skin menu (which can be activated by a right click).

- The **main window** and/or the **playbar** is where you can control MPlayer. The background of the window is an image. Various items can (and must) be placed in the window: *buttons*, *potmeters* (sliders) and *labels*. For every item, you must specify its position and size.

  A **button** has three states (pressed, released, disabled), thus its image must be divided into three parts vertically.

See the button item for details.

A **potmeter** (mainly used for the seek bar and volume/balance control) can have any number of phases by dividing its image into different parts below each other. See hpotmeter and potmeter for details.

Labels are a bit special: The characters needed to draw them are taken from an image file, and the characters in the image are described by a font description file. The latter is a plain text file which specifies the x,y position and size of each character in the image (the image file and its font description file form a font *together*). See dlabel and slabel for details.

Note

All images can have full transparency as described in the section about image formats. If the X server doesn't support the XShape extension, the parts marked transparent will be black. If you'd like to use this feature, the width of the main window's background image must be dividable by 8.

- The **subwindow** is where the movie appears. It can display a specified image if there is no movie loaded (it is quite boring to have an empty window :-)) **Note:** transparency is **not allowed** here.

- The **skin menu** is just a way to control MPlayer by means of menu entries. Two images are required for the menu: one of them is the base image that shows the menu in its normal state, the other one is used to display the selected entries. When you pop up the menu, the first image is shown. If you move the mouse over the menu entries, the currently selected entry is copied from the second image over the menu entry below the mouse pointer (the second image is never shown as a whole).

  A menu entry is defined by its position and size in the image (see the section about the skin menu for details).

There is an important thing not mentioned yet: For buttons, potmeters and menu entries to work, MPlayer must know what to do if they are clicked. This is done by messages (events). For these items you must define the messages to be generated when they are clicked.

B.1.2. Image formats

Images must be truecolor (24 or 32 bpp) PNGs.

In the main window and in the playbar (see below) you can use images with 'transparency': Regions filled with the color #FF00FF (magenta) are fully transparent when viewed by MPlayer. This means that you can even have shaped windows if your X server has the XShape extension.

B.1.3. Files

You need the following files to build a skin:

- The configuration file named skin tells MPlayer how to put different parts of the skin together and what to do if you click somewhere in the window.

- The background image for the main window.

- Images for the items in the main window (including one or more font description files needed to draw labels).

- The image to be displayed in the subwindow (optional).

- Two images for the skin menu (they are needed only if you want to create a menu).

With the exception of the skin configuration file, you can name the other files whatever you want (but note that font description files must have a `.fnt` extension).

B.2. The skin file

As mentioned above, this is the skin configuration file. It is line oriented; comment lines start with a '`;`' character at the beginning of the line (only spaces and tabs are allowed before the '`;`').

The file is made up of sections. Each section describes the skin for an application and has the following form:

```
section = section name
.
.
.
end
```

Currently there is only one application, so you need only one section: its name is **movieplayer**.

Within this section each window is described by a block of the following form:

```
window = window name
.
.
.
end
```

where *`window name`* can be one of these strings:

- **main** - for the main window
- **sub** - for the subwindow
- **menu** - for the skin menu
- **playbar** - playbar

(The sub and menu blocks are optional - you do not need to create a menu or decorate the subwindow.)

Within a window block, you can define each item for the window by a line in this form:

```
item = parameter
```

Where `item` is a string that identifies the type of the GUI item, `parameter` is a numeric or textual value (or a list of values separated by commas).

Putting the above together, the whole file looks something like this:

```
section = movieplayer
  window = main
  ; ... items for main window ...
  end

  window = sub
  ; ... items for subwindow ...
  end

  window = menu
  ; ... items for menu ...
  end

  window = playbar
  ; ... items for playbar ...
  end
end
```

The name of an image file must be given without leading directories - images are searched for in the `skins` directory. You may (but you need not) specify the extension of the file. If the file does not exist, MPlayer tries to load the file

<filename>.<ext>, where png and PNG are tried for <ext> (in this order). The first matching file will be used.

Here is an example to make this clear. Suppose that you have an image called main.png that you use for the main window:

```
base = main, -1, -1
```

MPlayer tries to load main, main.png, main.PNG files.

Finally some words about positioning. The main window and the subwindow can be placed in the different corners of the screen by giving X and Y coordinates. 0 is top or left, -1 is center and -2 is right or bottom, as shown in this illustration:

```
(0, 0)----(-1, 0)----(-2, 0)
  |          |          |
  |          |          |
(0,-1)----(-1,-1)----(-2,-1)
  |          |          |
  |          |          |
(0,-2)----(-1,-2)----(-2,-2)
```

B.2.1. Main window and playbar

Below is the list of entries that can be used in the 'window = main' ... 'end', and the 'window = playbar' ... 'end' blocks.

`base = image, X, Y`

Lets you specify the background image to be used for the main window. The window will appear at the given X,Y position on the screen The window will have the size of the image.

Note

These coordinates do not currently work for the display window.

Warning

Transparent regions in the image (colored #FF00FF) appear black on X servers without the XShape extension. The image's width must be dividable by 8.

`button = image, X, Y, width, height, message`

Place a button of width * height size at position X,Y. The specified message is generated when the button is clicked. The image given by image must have three parts below each other (according to the possible states of the button), like this:

```
+------------+
|  pressed   |
+------------+
|  released  |
+------------+
|  disabled  |
+------------+
```

`decoration = enable|disable`

Enable or disable window manager decoration of the main window. Default is **disable**.

Note

This doesn't work for the display window, there is no need to.

`hpotmeter = button, bwidth, bheight, phases, numphases, default, X, Y, width, height, message`

`vpotmeter = button, bwidth, bheight, phases, numphases, default, X, Y, width, height, message`

Place a horizontal (hpotmeter) or vertical (vpotmeter) potmeter of `width` * `height` size at position `X,Y`. The image can be divided into different parts for the different phases of the potmeter (for example, you can have a pot for volume control that turns from green to red while its value changes from the minimum to the maximum.). `hpotmeter` can have a button that can be dragged horizontally. The parameters are:

- `button` - the image to be used for the button (must have three parts below each other, like in case of button)
- `bwidth`, `bheight` - size of the button
- `phases` - the image to be used for the different phases of the hpotmeter. A special value of `NULL` can be used if you want no such image. The image must be divided into `numphases` parts vertically like this:

```
+------------+
|  phase #1  |
+------------+
|  phase #2  |
+------------+
     ...
+------------+
|  phase #n  |
+------------+
```

- `numphases` - number of phases stored in the `phases` image
- `default` - default value for hpotmeter (in the range `0` to `100`)
- X, Y - position for the hpotmeter
- `width`, `height` - width and height of the `hpotmeter`
- `message` - the message to be generated when the value of `hpotmeter` is changed

`potmeter = phases, numphases, default, X, Y, width, height, message`

A `hpotmeter` without a button. (I guess it is meant to be turned around, but it reacts to horizontal dragging only.) For the description of the parameters see hpotmeter. `phases` can be `NULL`, but it is quite useless, since you cannot see where the `potmeter` is set.

`font = fontfile, fontid`

Defines a font. `fontfile` is the name of a font description file with a `.fnt` extension (do not specify the extension here). `fontid` is used to refer to the font (see dlabel and slabel). Up to 25 fonts can be defined.

`slabel = X, Y, fontid, "text"`

Place a static label at the position `X,Y`. `text` is displayed using the font identified by `fontid`. The text is just a raw string ($x variables do not work) that must be enclosed between double quotes (but the " character cannot be part of the text). The label is displayed using the font identified by `fontid`.

`dlabel = X, Y, length, align, fontid, "text"`

Place a dynamic label at the position `X,Y`. The label is called dynamic because its text is refreshed periodically. The maximum length of the label is given by `length` (its height is the height of a character). If the text to be displayed is wider than that, it will be scrolled, otherwise it is aligned within the specified space by the value of the `align` parameter: `0` is for right, `1` is for center, `2` is for left.

The text to be displayed is given by `text`: It must be written between double quotes (but the " character cannot be part of the text). The label is displayed using the font identified by `fontid`. You can use the following variables in the text:

Variable	Meaning
$1	play time in *hh:mm:ss* format
$2	play time in *mmmm:ss* format
$3	play time in *hh* format (hours)
$4	play time in *mm* format (minutes)
$5	play time in *ss* format (seconds)
$6	movie length in *hh:mm:ss* format
$7	movie length in *mmmm:ss* format
$8	play time in *h:mm:ss* format
$v	volume in *xxx.xx*% format
$V	volume in *xxx.xx* format
$b	balance in *xxx.xx*% format
$B	balance in *xxx.xx* format
$$	the $ character
$a	a character according to the audio type (none: `n`, mono: `m`, stereo: `t`)
$t	track number (in playlist)
$o	filename
$f	filename in lower case
$F	filename in upper case
$T	a character according to the stream type (file: `f`, Video CD: `v`, DVD: `d`, URL: `u`)
$p	the **p** character (if a movie is playing and the font has the **p** character)
$s	the **s** character (if the movie is stopped and the font has the **s** character)
$e	the **e** character (if playback is paused and the font has the **e** character)
$x	movie width
$y	movie height
$C	name of the codec used

Note

The `$a, $T, $p, $s` and `$e` variables all return characters that should be displayed as special symbols (for example, **e** is for the pause symbol that usually looks something like ||). You should have a font for normal characters and a different font for symbols. See the section about symbols for more information.

B.2.2. Subwindow

The following entries can be used in the '`window = sub`'. . .'`end`' block.

`base = image, X, Y, width, height`

The image to be displayed in the window. The window will appear at the given `X,Y` position on the screen (`0,0` is the top left corner). You can specify `-1` for center and `-2` for right (`X`) and bottom (`Y`). The window will be as large as the image. `width` and `height` denote the size of the window; they are optional (if they are missing, the window is the same size as the image).

`background = R, G, B`

Lets you set the background color. It is useful if the image is smaller than the window. `R`, `G` and `B` specifies the red, green and blue component of the color (each of them is a decimal number from 0 to 255).

B.2.3. Skin menu

As mentioned earlier, the menu is displayed using two images. Normal menu entries are taken from the image specified by the `base` item, while the currently selected entry is taken from the image specified by the `selected` item. You must define the position and size of each menu entry through the menu item.

The following entries can be used in the ’`window = menu`’. . .’`end`’ block.

`base = image`

The image for normal menu entries.

`selected = image`

The image showing the menu with all entries selected.

`menu = X, Y, width, height, message`

Defines the `X,Y` position and the size of a menu entry in the image. `message` is the message to be generated when the mouse button is released over the entry.

B.3. Fonts

As mentioned in the section about the parts of a skin, a font is defined by an image and a description file. You can place the characters anywhere in the image, but make sure that their position and size is given in the description file exactly.

The font description file (with `.fnt` extension) can have comment lines starting with ’`;`’. The file must have a line in the form

```
image = image
```

Where *`image`* is the name of the image file to be used for the font (you do not have to specify the extension).

```
"char" = X, Y, width, height
```

Here `X` and `Y` specify the position of the `char` character in the image (`0,0` is the upper left corner). `width` and `height` are the dimensions of the character in pixels.

This example defines the A, B, C characters using `font.png`.

```
; Can be "font" instead of "font.png".
image = font.png

; Three characters are enough for demonstration purposes :-)
"A" =  0,0, 7,13
"B" =  7,0, 7,13
"C" = 14,0, 7,13
```

B.3.1. Symbols

Some characters have special meanings when returned by some of the variables used in dlabel. These characters are meant to be shown as symbols so that things like a nice DVD logo can be displayed instead of the character ’d’ for a DVD stream.

The following table lists all the characters that can be used to display symbols (and thus require a different font).

Character	Symbol
p	play
s	stop
e	pause
n	no sound
m	mono sound
t	stereo sound
f	stream is a file
v	stream is a Video CD
d	stream is a DVD
u	stream is a URL

B.4. GUI messages

These are the messages that can be generated by buttons, potmeters and menu entries.

Playback control:

evNext

Jump to next track in the playlist.

evPause

Forms a switch together with `evPlaySwitchToPause`. They can be used to have a common play/pause button. Both messages should be assigned to buttons displayed at the very same position in the window. This message pauses playing and the image for the `evPlaySwitchToPause` button is displayed (to indicate that the button can be pressed to continue playing).

evPlay

Start playing.

evPlaySwitchToPause

The opposite of `evPauseSwitchToPlay`. This message starts playing and the image for the `evPauseSwitchToPlay` button is displayed (to indicate that the button can be pressed to pause playing).

evPrev

Jump to previous track in the playlist.

evStop

Stop playing.

Seeking:

evBackward10sec

Seek backward 10 seconds.

evBackward1min

Seek backward 1 minute.

evBackward10min

Seek backward 10 minutes.

evForward10sec

Seek forward 10 seconds.

evForward1min

Seek forward 1 minute.

evForward10min

Seek forward 10 minutes.

evSetMoviePosition

Seek to position (can be used by a potmeter; the relative value (0-100%) of the potmeter is used).

Video control:

evHalfSize

Set the movie window to half size.

evDoubleSize

Set the movie window to double size.

evFullScreen

Switch fullscreen mode on/off.

evNormalSize

Set the movie window to its normal size.

Audio control:

evDecAudioBufDelay

Decrease audio buffer delay.

evDecBalance

Decrease balance.

evDecVolume

Decrease volume.

evIncAudioBufDelay

Increase audio buffer delay.

evIncBalance

Increase balance.

evIncVolume

Increase volume.

evMute

Mute/unmute the sound.

evSetBalance

Set balance (can be used by a potmeter; the relative value (0-100%) of the potmeter is used).

evSetVolume

Set volume (can be used by a potmeter; the relative value (0-100%) of the potmeter is used).

Miscellaneous:

evAbout

Open the about window.

evDropSubtitle

Disables the currently used subtitle.

evEqualizer

Turn the equalizer on/off.

evExit

Quit the program.

evIconify

Iconify the window.

evLoad

Load a file (by opening a file browser window, where you can choose a file).

evLoadPlay

Does the same as `evLoad`, but it automatically starts playing after the file is loaded.

evLoadSubtitle

Loads a subtitle file (with the file selector).

evLoadAudioFile

Loads an audio file (with the file selector).

evNone

Empty message, it has no effect (except maybe in Subversion versions :-)).

evPlaylist

Open/close the playlist window.

evPlayDVD

Tries to open the disc in the given DVD-ROM drive.

evPlayVCD

Tries to open the disc in the given CD-ROM drive.

evPreferences

Open the preferences window.

evSetAspect

Sets displayed image aspect.

evSetURL

Displays the URL dialog window.

evSkinBrowser

Open the skin browser window.

B.5. Creating quality skins

So you have read up on creating skins for the MPlayer GUI, done your best with the Gimp and wish to submit your skin to us? Read on for some guidelines to avoid common mistakes and produce a high quality skin.

We want skins that we add to our repository to conform to certain quality standards. There are also a number of things that you can do to make our lives easier.

As an example you can look at the `Blue` skin, it satisfies all the criteria listed below since version 1.5.

- Each skin should come with a `README` file that contains information about you, the author, copyright and license notices and anything else you wish to add. If you wish to have a changelog, this file is a good place.
- There should be a file `VERSION` with nothing more than the version number of the skin on a single line (e.g. 1.0).
- Horizontal and vertical controls (sliders like volume or position) should have the center of the knob properly centered on the middle of the slider. It should be possible to move the knob to both ends of the slider, but not past it.
- Skin elements should have the right sizes declared in the skin file. If this is not the case you can click outside of e.g. a button and still trigger it or click inside its area and not trigger it.
- The `skin` file should be prettyprinted and not contain tabs. Prettyprinted means that the numbers should line up neatly in columns.