mkvmerge — Merge multimedia streams into a Matroska™ file

Synopsis

`mkvmerge` [global options] {-o out} [options1] {file1} [[options2] {file2}] [@optionsfile]

Description

This program takes the input from several media files and joins their streams (all of them or just a selection) into a Matroska™ file; see the Matroska™ website.

Global options:

`-v`, `--verbose`

Increase verbosity.

`-q`, `--quiet`

Suppress status output.

`-o`, `--output` *`file-name`*

Write to the file *`file-name`*. If splitting is used then this parameter is treated a bit differently. See the explanation for the `--split` option for details.

`-w`, `--webm`

Create a WebM compliant file. This is also turned on if the output file name's extension is "webm". This mode enforces several restrictions. The only allowed codecs are VP8 video and Vorbis audio tracks. Neither chapters nor tags are allowed. The DocType header item is changed to "webm".

`--title` *`title`*

Sets the general title for the output file, e.g. the movie name.

`--tags` *`file-name`*

Read global tags from the XML file *file-name*. See the section about tags below for details.

`--default-language` *language-code*

Sets the default language code that will be used for all tracks unless overwritten with the `--language` option. The default language code is '`und`' for 'undefined'.

Segment info handling: (global options)

`--segmentinfo` *filename.xml*

Read segment information from a XML file. This file can contain the segment family UID, segment UID, previous and next segment UID elements. An example file and a DTD are included in the MKVToolNix distribution.

`--segment-uid` *SID1,SID2,...*

Sets the segment UIDs to use. This is a comma-separated list of 128bit segment UIDs in the usual UID form: hex numbers with or without the "0x" prefix, with or without spaces, exactly 32 digits.

Each file created contains one segment, and each segment has one segment UID. If more segment UIDs are specified than segments are created then the surplus UIDs are ignored. If fewer UIDs are specified than segments are created then random UIDs will be created for them.

Chapter and tag handling: (global options)

`--chapter-language` *language-code*

Sets the ISO639-2 language code that is written for each chapter entry. Defaults to '`eng`'. See the section about chapters below for details.

This option can be used both for simple chapter files and for source files that contain chapters but no information about the chapters' language, e.g. MP4 and OGM files.

`--chapter-charset` *character-set*

Sets the character set that is used for the conversion to UTF-8 for simple chapter files. See the section about text files and character sets for an explanation how mkvmerge(1) converts between character sets.

This switch does also apply to chapters that are copied from certain container types, e.g. Ogg/OGM and MP4 files. See the section about chapters below for details.

`--cue-chapter-name-format` *format*

mkvmerge(1) supports reading CUE sheets for audio files as the input for chapters. CUE sheets usually contain the entries `PERFORMER` and `TITLE` for each index entry. mkvmerge(1) uses these two strings in order to construct the chapter name. With this option the format used for this name can be set.

If this option is not given then mkvmerge(1) defaults to the format '`%p - %t`' (the performer, followed by a space, a dash, another space and the title).

If the format is given then everything except the following meta characters is copied as-is, and the meta characters are replaced like this:

- *`%p`* is replaced by the current entry's `PERFORMER` string,
- *`%t`* is replaced by the current entry's `TITLE` string,
- *`%n`* is replaced by the current track number and
- *`%N`* is replaced by the current track number padded with a leading zero if it is < 10.

`--chapters` *file-name*

Read chapter information from the file *`file-name`*. See the section about chapters below for details.

`--global-tags` *file-name*

Read global tags from the file *`file-name`*. See the section about tags below for details.

General output control (advanced global options):

`--track-order` *FID1:TID1,FID2:TID2,...*

This option changes the order in which the tracks for an input file are created. The argument is a comma separated list of pairs IDs. Each pair contains first the file ID (*`FID1`*) which is simply the number of the file on the command line starting at 0. The second is a track ID (*`TID1`*) from that file. If some track IDs

are omitted then those tracks are created after the ones given with this option have been created.

`--cluster-length` *spec*

Limit the number of data blocks or the duration of data in each cluster. The *spec* parameter can either be a number *n* without a unit or a number *d* postfixed with '`ms`'.

If no unit is used then mkvmerge(1) will put at most *n* data blocks into each cluster. The maximum number of blocks is 65535.

If the number *d* is postfixed with '`ms`' then mkvmerge(1) puts at most *d* milliseconds of data into each cluster. The minimum for *d* is '`100ms`', and the maximum is '`32000ms`'.

mkvmerge(1) defaults to putting at most 65535 data blocks and 5000ms of data into a cluster.

Programs trying to find a certain frame can only seek directly to a cluster and have to read the whole cluster afterwards. Therefore creating larger clusters may lead to imprecise or slow seeking.

`--no-cues`

Tells mkvmerge(1) not to create and write the cue data which can be compared to an index in an AVI. Matroska™ files can be played back without the cue data, but seeking will probably be imprecise and slower. Use this only if you're really desperate for space or for testing purposes. See also option `--cues` which can be specified for each input file.

`--clusters-in-meta-seek`

Tells mkvmerge(1) to create a meta seek element at the end of the file containing all clusters. See also the section about the Matroska™ file layout.

`--disable-lacing`

Disables lacing for all tracks. This will increase the file's size, especially if there are many audio tracks. This option is not intended for everyday use.

`--enable-durations`

Write durations for all blocks. This will increase file size and does not offer any additional value for players at the moment.

`--timecode-scale` *factor*

Forces the timecode scale factor to *factor*. Valid values are in the range `1000`..`10000000` or the special value `-1`.

Normally mkvmerge(1) will use a value of `1000000` which means that timecodes and durations will have a precision of 1ms. For files that will not contain a video track but at least one audio track mkvmerge(1) will automatically chose a timecode scale factor so that all timecodes and durations have a precision of one audio sample. This causes bigger overhead but allows precise seeking and extraction.

If the special value `-1` is used then mkvmerge(1) will use sample precision even if a video track is present.

File splitting, linking and appending (more global options):

`--split` *specification*

Splits the output file after a given size or a given time. Please note that tracks can only be split right before a key frame. Due to buffering mkvmerge(1) will split right before the next key frame after the split point has been reached. Therefore the split point may be a bit off from what the user has specified.

At the moment mkvmerge(1) supports three different modes.

1. Splitting by size.

   Syntax: `--split` [size:]*d*[k|m|g]

   Examples: `--split size:700m` or `--split 150000000`

   The parameter *d* may end with '`k`', '`m`' or '`g`' to indicate that the size is in KB, MB or GB respectively. Otherwise a size in Bytes is assumed. After the current output file has reached this size limit a new one will be started.

   The '`size:`' prefix may be omitted for compatibility reasons.

2. Splitting after a duration.

Syntax: `--split` [duration:]*`HH:MM:SS.nnnnnnnnn`*|*`d`*s

Examples: `--split duration:00:60:00.000` or `--split 3600s`

The parameter must either have the form *`HH:MM:SS.nnnnnnnnn`* for specifying the duration in up to nano-second precision or be a number *`d`* followed by the letter '`s`' for the duration in seconds. *`HH`* is the number of hours, *`MM`*the number of minutes, *`SS`* the number of seconds and *`nnnnnnnnn`* the number of nanoseconds. Both the number of hours and the number of nanoseconds can be omitted. There can be up to nine digits after the decimal point. After the duration of the contents in the current output has reached this limit a new output file will be started.

The '`duration:`' prefix may be omitted for compatibility reasons.

3. Splitting after specific timecodes.

   Syntax: `--split` timecodes:*`A`*[,*`B`*[,*`C`*...]]

   Example: `--split timecodes:00:45:00.000,01:20:00.250,6300s`

   The parameters *`A`*, *`B`*, *`C`* etc must all have the same format as the ones used for the duration (see above). The list of timecodes is separated by commas. After the input stream has reached the current split point's timecode a new file is created. Then the next split point given in this list is used.

   The '`timecodes:`' prefix must not be omitted.

For this splitting mode the output filename is treated differently than for the normal operation. It may contain a `printf` like expression '`%d`' including an optional field width, e.g. '`%02d`'. If it does then the current file number will be formatted appropriately and inserted at that point in the filename. If there is no such pattern then a pattern of '`-%03d`' is assumed right before the file's extension: '`-o output.mkv`' would result in '`output-001.mkv`' and so on. If there's no extension then '`-%03d`' will be appended to the name.

`--link`

Link files to one another when splitting the output file. See the section on file linking below for details.

`--link-to-previous` *`segment-UID`*

Links the first output file to the segment with the segment UID given by the *segment-UID* parameter. See the section on file linking below for details.

`--link-to-next` *segment-UID*

Links the last output file to the segment with the segment UID given by the *segment-UID* parameter. See the section on file linking below for details.

`--append-mode` *mode*

Determines how timecodes are calculated when appending files. The parameter *mode* can have two values: '`file`' which is also the default and '`track`'.

When mkvmerge appends a track (called '`track2_1`' from now on) from a second file (called '`file2`') to a track (called '`track1_1`') from the first file (called '`file1`') then it has to offset all timecodes for '`track2_1`' by an amount. For '`file`' mode this amount is the highest timecode encountered in '`file1`' even if that timecode was from a different track than '`track1_1`'. In track mode the offset is the highest timecode of '`track1_1`'.

Unfortunately mkvmerge cannot detect which mode to use reliably. Therefore it defaults to '`file`' mode. '`file`' mode usually works better for files that have been created independently of each other; e.g. when appending AVI orMP4 files. '`track`' mode may work better for sources that are essentially just parts of one big file, e.g. for VOB and EVO files.

Subtitle tracks are always treated as if '`file`' mode were active even if '`track`' mode actually is.

`--append-to` *SFID1:STID1:DFID1:DTID1[,...]*

This option controls to which track another track is appended. Each spec contains four IDs: a file ID, a track ID, a second file ID and a second track ID. The first pair, "source file ID" and "source track ID", identifies the track that is to be appended. The second pair, "destination file ID" and "destination track ID", identifies the track the first one is appended to.

If this option has been omitted then a standard mapping is used. This standard mapping appends each track from the current file to a track from the previous file with the same track ID. This allows for easy appending if a movie has been split into two parts and both file have the same number of tracks and track IDs with the command **mkvmerge -o output.mkv part1.mkv +part2.mkv**.

+

A single '+' causes the next file to be appended instead of added. The '+' can also be put in front of the next file name. Therefore the following two commands are equivalent:

```
$ mkvmerge -o full.mkv file1.mkv + file2.mkv
$ mkvmerge -o full.mkv file1.mkv +file2.mkv
```

+

Normally mkvmerge looks for files in the same directory as an input file that have the same base name and only differ in their running number (e.g. 'VTS_01_1.VOB', 'VTS_01_2.VOB', 'VTS_01_3.VOB' etc). This option, a single '=', causes mkvmerge not to look for those additional files.

The '=' can also be put in front of the next file name. Therefore the following two commands are equivalent:

```
$ mkvmerge -o full.mkv = file1.mkv
$ mkvmerge -o full.mkv =file1.mkv
```

Attachment support (more global options):

`--attachment-description` *`description`*

Plain text description of the following attachment. Applies to the next `--attach-file` or `--attach-file-once` option.

`--attachment-mime-type` *`MIME type`*

MIME type of the following attachment. Applies to the next `--attach-file` or `--attach-file-once` option. A list of officially recognized MIME types can be found e.g. at the IANA homepage. The MIME type is mandatory for an attachment.

`--attachment-name` *`name`*

Sets the name that will be stored in the output file for this attachment. If this option is not given then the name will be derived from the file name of the attachment as given with the `--attach-file` or the `--attach-file-once`option.

`--attach-file` *`file-name`*, `--attach-file-once` *`file-name`*

Creates a file attachment inside the Matroska™ file. The MIME type must have been set before this option can used. The difference between the two forms is that during splitting the files attached with `--attach-file` are attached to all output files while the ones attached with `--attach-file-once` are only attached to the first file created. If splitting is not used then both do the same.

mkvextract(1) can be used to extract attached files from a Matroska™ file.

Options that can be used for each input file:

`-a`, `--audio-tracks` *n,m,...*

Copy the audio tracks *n*, *m* etc. The numbers are track IDs which can be obtained with the `--identify` switch. They're not simply the track numbers (see section track IDs). Default: copy all audio tracks.

`-d`, `--video-tracks` *n,m,...*

Copy the video tracks *n*, *m* etc. The numbers are track IDs which can be obtained with the `--identify` switch. They're not simply the track numbers (see section track IDs). Default: copy all video tracks.

`-s`, `--subtitle-tracks` *n,m,...*

Copy the subtitle tracks *n*, *m* etc. The numbers are track IDs which can be obtained with the `--identify` switch. They're not simply the track numbers (see section track IDs). Default: copy all subtitle tracks.

`-b`, `--button-tracks` *n,m,...*

Copy the button tracks *n*, *m* etc. The numbers are track IDs which can be obtained with the `--identify` switch. They're not simply the track numbers (see section track IDs). Default: copy all button tracks.

`--track-tags` *n,m,...*

Copy the tags for tracks *n*, *m* etc. The numbers are track IDs which can be obtained with the `--identify` switch (see section track IDs). They're not simply the track numbers. Default: copy tags for all tracks.

`-m`, `--attachments` *n[:all|first],m[:all|first],...*

Copy the attachments with the IDs *n*, *m* etc to all or only the first output file. Each ID can be followed by either ':all' (which is the default if neither is entered) or ':first'. If splitting is active then those attachments whose IDs are specified with ':all' are copied to all of the resulting output files while the others are only copied into the first output file. If splitting is not active then both variants have the same effect.

The default is to copy all attachments to all output files.

-A, **--no-audio**

Don't copy any audio track from this file.

-D, **--no-video**

Don't copy any video track from this file.

-S, **--no-subtitles**

Don't copy any subtitle track from this file.

-B, **--no-buttons**

Don't copy any button track from this file.

-T, **--no-track-tags**

Don't copy any track specific tags from this file.

--no-chapters

Don't copy chapters from this file.

-M, **--no-attachments**

Don't copy attachments from this file.

--no-global-tags

Don't copy global tags from this file.

--chapter-charset *character-set*

Sets the charset that is used for the conversion to UTF-8 for chapter information contained in the source file. See the section about text files and character sets for an explanation how mkvmerge(1) converts between character sets.

`--chapter-language` *language-code*

Sets the ISO639-2 language code that is written for each chapter entry. This option can be used for source files that contain chapters but no information about the chapters' languages, e.g. for MP4 and OGM files.

`-y`, `--sync` *TID:d[,o[/p]]*

Adjust the timecodes of the track with the id *TID* by *d* ms. The track IDs are the same as the ones given with `--identify` (see section track IDs).

o/*p*: adjust the timestamps by *o*/*p* to fix linear drifts. *p* defaults to 1 if omitted. Both *o* and *p* can be floating point numbers.

Defaults: no manual sync correction (which is the same as *d* = `0` and *o*/*p* = `1.0`).

This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

`--cues` *TID:none|iframes|all*

Controls for which tracks cue (index) entries are created for the given track (see section track IDs). '`none`' inhibits the creation of cue entries. For '`iframes`' only blocks with no backward or forward references (= I frames in video tracks) are put into the cue sheet. '`all`' causes mkvmerge(1) to create cue entries for all blocks which will make the file very big.

The default is '`iframes`' for video tracks and '`none`' for all others. See also option `--no-cues` which inhibits the creation of cue entries regardless of the `--cues` options used.

This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

`--default-track` *TID[:bool]*

Sets the 'default' flag for the given track (see section track IDs) if the optional argument *bool* is not present. If the user does not explicitly select a track himself then the player should prefer the track that has his 'default' flag set. Only one track of each kind (audio, video, subtitles, buttons) can have his 'default' flag set. If the user wants no track to have the default track flag set then he has to set *bool* to `0` for all tracks.

This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

`--forced-track` *TID[:bool]*

Sets the 'forced' flag for the given track (see section track IDs) if the optional argument *bool* is not present. A player must play all tracks for which this flag is set to `1`.

This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

`--blockadd` *TID:level*

Keep only the `BlockAdditions` up to the level *level* for the given track. The default is to keep all levels. This option only affects certain kinds of codecs like WAVPACK4.

`--track-name` *TID:name*

Sets the track name for the given track (see section track IDs) to *name*.

`--language` *TID:language*

Sets the language for the given track (see section track IDs). Both ISO639-2 language codes and ISO639-1 country codes are allowed. The country codes will be converted to language codes automatically. All languages including their ISO639-2 codes can be listed with the `--list-languages` option.

This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

`-t`, `--tags` *TID:file-name*

Read tags for the track with the number *TID* from the file *file-name*. See the section about tags below for details.

`--aac-is-sbr` *TID[:0|1]*

Tells mkvmerge(1) that the track with the ID *TID* is SBR AAC (also known as HE-AAC or AAC+). This options is needed if a) the source file is an AAC file (*not* for a Matroska™ file) and b) the AAC file contains SBR AACdata. The reason for this switch is that it is technically impossible to automatically tell normal AAC data from SBR AAC data without decoding a complete AAC frame. As there are several patent issues with AAC decodersmkvmerge(1) will never contain this decoding stage. So for SBR AAC files this switch is mandatory. The resulting file might not play back correctly or even not at all if the switch was omitted.

If the source file is a Matroska™ file then the `CodecID` should be enough to detect SBR AAC. However, if the `CodecID` is wrong then this switch can be used to correct that.

If mkvmerge wrongfully detects that an AAC file is SBR then you can add '`:0`' to the track ID.

`--timecodes` *TID:file-name*

Read the timecodes to be used for the specific track ID from *file-name*. These timecodes forcefully override the timecodes that mkvmerge(1) normally calculates. Read the section about external timecode files.

`--default-duration` *TID:x*

Forces the default duration of a given track to the specified value. Also modifies the track's timecodes to match the default duration. The argument *x* must be postfixed with '`s`', '`ms`', '`us`', '`ns`' or '`fps`' to specify the default duration in seconds, milliseconds, microseconds, nanoseconds or 'frames per second' respectively. The number *x* itself can be a floating point number or a fraction.

If the default duration is not forced then mkvmerge will try to derive the track's default duration from the container and/or codec used. One case in which this option is of use is when adding *AVC/h.264* elementary streams because these do not contain information about their number of frames or a default duration

for each frame. For such files mkvmerge(1) will assume a default duration of '`25fps`' unless overridden.

This option can also be used to change the FPS of video tracks without having to use an external timecode file.

`--nalu-size-length` *`TID:n`*

Forces the NALU size length to *`n`* bytes. This parameter is only used if the *AVC/h.264* elementary stream packetizer is used. If left out it defaults to 4 bytes, but there are files that contain frames or slices that are all smaller than 65536 bytes. For such files you can use this parameter and decrease the size to 2.

Options that only apply to video tracks:

`-f`, `--fourcc` *`TID:FourCC`*

Forces the `FourCC` to the specified value. Works only for video tracks in the '*MS compatibility mode*'.

`--display-dimensions` *`TID:widthxheight`*

Matroska™ files contain two values that set the display properties that a player should scale the image on playback to: display width and display height. These values can be set with this option, e.g. '`1:640x480`'.

Another way to specify the values is to use the `--aspect-ratio` or the `--aspect-ratio-factor` option (see below). These options are mutually exclusive.

`--aspect-ratio` *`TID:ratio|width/height`*

Matroska™ files contain two values that set the display properties that a player should scale the image on playback to: display width and display height. With this option mkvmerge(1) will automatically calculate the display width and display height based on the image's original width and height and the aspect ratio given with this option. The ratio can be given either as a floating point number *`ratio`* or as a fraction '*`width`*/*`height`*', e.g. '`16/9`'.

Another way to specify the values is to use the `--aspect-ratio-factor` or `--display-dimensions` options (see above and below). These options are mutually exclusive.

`--aspect-ratio-factor` *`TID:factor|n/d`*

Another way to set the aspect ratio is to specify a *`factor`*. The original aspect ratio is first multiplied with this *`factor`* and used as the target aspect ratio afterwards.

Another way to specify the values is to use the `--aspect-ratio` or `--display-dimensions` options (see above). These options are mutually exclusive.

`--cropping` *`TID:left,top,right,bottom`*

Sets the pixel cropping parameters of a video track to the given values.

`--stereo-mode` *`TID:n|keyword`*

Sets the stereo mode for the video track with the track ID *`TID`*. The mode can either be a number *`n`* between `0` and `3` or one of the keywords '`none`' (same as *`n`*=`0`), '`right`' (same as *`n`*=`1`), '`left`' (same as *`n`*=`2`) or '`both`' (same as*`n`*=`3`).

`--compression` *`TID:method`*

Selects the compression method to be used for the VobSub track. Note that the player also has to support this method. Valid values are '`none`', '`zlib`', '`lzo`'/'`lzo1x`', '`bz2`'/'`bzlib`' and '`mpeg4_p2`'/'`mpeg4p2`'. The values '`lzo`'/'`lzo1x`' and '`bz2`'/'`bzlib`' are only available if mkvmerge(1) has been compiled with support for the liblzo™ respectively bzlib™ compression libraries.

The compression method '`mpeg4_p2`'/'`mpeg4p2`' is a special compression method called '*header removal*' that is only available for MPEG4 part 2 video tracks. The other methods are general compression methods that can be used with any type of track.

The default is '`zlib`' compression. This compression method is also the one that most if not all playback applications support. Support for other compression methods other than '`none`' is not assured.

Options that only apply to text subtitle tracks:

`--sub-charset` *`TID:character-set`*

Sets the character set for the conversion to UTF-8 for UTF-8 subtitles for the given track ID. If not specified the charset will be derived from the current

locale settings. Note that a charset is not needed for subtitles read fromMatroska™ files or from Kate streams, as these are always stored in UTF-8. See the section about text files and character sets for an explanation how mkvmerge(1) converts between character sets.

This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

Other options:

`-i`, `--identify` *file-name*

Will let mkvmerge(1) probe the single file and report its type, the tracks contained in the file and their track IDs. If this option is used then the only other option allowed is the filename.

`-l`, `--list-types`

Lists supported input file types.

`--list-languages`

Lists all languages and their ISO639-2 code which can be used with the `--language` option.

`--priority` *priority*

Sets the process priority that mkvmerge(1) runs with. Valid values are '`lowest`', '`lower`', '`normal`', '`higher`' and '`highest`'. If nothing is given then '`normal`' is used. On Unix like systems mkvmerge(1) will use the nice(2) function. Therefore only the super user can use '`higher`' and '`highest`'. On Windows all values are useable for every user.

`--command-line-charset` *character-set*

Sets the character set to convert strings given on the command line from. It defaults to the character set given by system's current locale. This settings applies to arguments of the following options: `--title`, `--track-name` and `--attachment-description`.

`--output-charset` *character-set*

Sets the character set to which strings are converted that are to be output. It defaults to the character set given by system's current locale.

`-r`, `--redirect-output` *`file-name`*

Writes all messages to the file *`file-name`* instead of to the console. While this can be done easily with output redirection there are cases in which this option is needed: when the terminal reinterprets the output before writing it to a file. The character set set with `--output-charset` is honored.

`--ui-language` *`code`*

Forces the translations for the language *`code`* to be used (e.g. '`de_DE`' for the German translations). It is preferable to use the environment variables `LANG`, `LC_MESSAGES` and `LC_ALL` though. Entering '`list`' as the *`code`* will causemkvmerge(1) to output a list of available translations.

`@`options-file

Reads additional command line arguments from the file *`options-file`*. Lines whose first non-whitespace character is a hash mark ('`#`') are treated as comments and ignored. White spaces at the start and end of a line will be stripped. Each line must contain exactly one option. There is no meta character escaping.

The command line '**mkvmerge -o "my file.mkv" -A "a movie.avi" sound.ogg**' could be converted into the following option file:

```
# Write to the file "my file.mkv".
-o
my file.mkv
# Only take the video from "a movie.avi".
-A
a movie.avi
sound.ogg
```

`--capabilities`

Lists information about optional features that have been compiled in and exit. The first line output will be the version information. All following lines contain exactly one word whose presence indicates that the feature has been compiled in. These features are:

- '`BZ2`' -- the bzlib™ compression library. Affects the available compression methods for the `--compression` option.

- 'LZO' -- the lzo™ compression library. Affects the available compression methods for the --compression option.
- 'FLAC' -- reading raw FLAC files and handling FLAC tracks in other containers, e.g. Ogg™ or Matroska™.

-h, --help

Show usage information and exit.

-V, --version

Show version information and exit.

Usage

For each file the user can select which tracks mkvmerge(1) should take. They are all put into the file specified with -o. A list of known (and supported) source formats can be obtained with the -l option.

Examples

Let's assume you have a file called MyMovie.avi and the audio track in a separate file, e.g. 'MyMovie.wav'. First you want to encode the audio to OggVorbis™:

```
$ oggenc -q4 -oMyMovie.ogg MyMovie.wav
```

After a couple of minutes you can join video and audio:

```
$ mkvmerge -o MyMovie-with-sound.mkv MyMovie.avi MyMovie.ogg
```

If your AVI already contains an audio track then it will be copied as well (if mkvmerge(1) supports the audio format). To avoid that simply do

```
$ mkvmerge -o MyMovie-with-sound.mkv -A MyMovie.avi MyMovie.ogg
```

After some minutes of consideration you rip another audio track, e.g. the director's comments or another language to 'MyMovie-add-audio.wav'. Encode it again and join it up with the other file:

```
$ oggenc -q4 -oMyMovie-add-audio.ogg MyMovie-add-audio.wav
$ mkvmerge -o MM-complete.mkv MyMovie-with-sound.mkv MyMovie-add-audio.ogg
```

The same result can be achieved with

```
$ mkvmerge -o MM-complete.mkv -A MyMovie.avi MyMovie.ogg MyMovie-add-
audio.ogg
```

Now fire up mplayer™ and enjoy. If you have multiple audio tracks (or even video tracks) then you can tell mplayer™ which track to play with the '`-vid`' and '`-aid`' options. These are 0-based and do not distinguish between video and audio.

If you need an audio track synchronized you can do that easily. First find out which track ID the Vorbis track has with

```
$ mkvmerge --identify outofsync.ogg
```

Now you can use that ID in the following command line:

```
$ mkvmerge -o goodsync.mkv -A source.avi -y 12345:200 outofsync.ogg
```

This would add 200ms of silence at the beginning of the audio track with the ID `12345` taken from '`outofsync.ogg`'.

Some movies start synced correctly but slowly drift out of sync. For these kind of movies you can specify a delay factor that is applied to all timestamps -- no data is added or removed. So if you make that factor too big or too small you'll get bad results. An example is that an episode I transcoded was `0.2` seconds out of sync at the end of the movie which was `77340` frames long. At `29.97fps` `0.2` seconds correspond to approx. `6` frames. So I did

```
$ mkvmerge -o goodsync.mkv -y 23456:0,77346/77340 outofsync.mkv
```

The result was fine.

The sync options can also be used for subtitles in the same manner.

For text subtitles you can either use some Windows software (like SubRipper™) or the subrip™ package found in transcode(1)'s sources in the '`contrib/subrip`' directory. The general process is:

1. extract a raw subtitle stream from the source:

   ```
   $ tccat -i /path/to/copied/dvd/ -T 1 -L | tcextract -x ps1 -t vob -a
   0x20 | subtitle2pgm -o mymovie
   ```

2. convert the resulting PGM images to text with gocr:

   ```
   $ pgm2txt mymovie
   ```

3. spell-check the resulting text files:

```
$ ispell -d american *txt
```

4. convert the text files to a SRT file:

```
$ srttool -s -w -i mymovie.srtx -o mymovie.srt
```

The resulting file can be used as another input file for mkvmerge(1):

```
$ mkvmerge -o mymovie.mkv mymovie.avi mymovie.srt
```

If you want to specify the language for a given track then this is easily done. First find out the ISO639-2 code for your language. mkvmerge(1) can list all of those codes for you:

```
$ mkvmerge --list-languages
```

Search the list for the languages you need. Let's assume you have put two audio tracks into a Matroska™ file and want to set their language codes and that their track IDs are 2 and 3. This can be done with

```
$ mkvmerge -o with-lang-codes.mkv --language 2:ger --language 3:dut without-
lang-codes.mkv
```

As you can see you can use the `--language` switch multiple times.

Maybe you'd also like to have the player use the Dutch language as the default language. You also have extra subtitles, e.g. in English and French, and want to have the player display the French ones by default. This can be done with

```
$ mkvmerge -o with-lang-codes.mkv --language 2:ger --language 3:dut --
default-track 3 without-lang-codes.mkv --language 0:eng english.srt --
default-track 0 --language 0:fre french.srt
```

If you do not see the language or default track flags that you've specified in mkvinfo(1)'s output then please read the section about default values.

Track IDs

Some of the options for mkvmerge(1) need a track ID to specify which track they should be applied to. Those track IDs are printed by the readers when demuxing the current input file, or if mkvmerge(1) is called with the `--identify`option. An example for such output:

```
$ mkvmerge -i v.mkv
File 'v.mkv': container: Matroska™
Track ID 1: video (V_MS/VFW/FOURCC, DIV3)
Track ID 2: audio (A_MPEG/L3)
```

Track IDs are assigned like this:

- AVI files: The video track has the ID 0. The audio tracks get IDs in ascending order starting at 1.
- AAC, AC3, MP3, SRT and WAV files: The one 'track' in that file gets the ID 0.
- Ogg/OGM files: The track IDs are assigned in order the tracks are found in the file starting at 0.
- Matroska™ files: The track's ID is the track number as reported by mkvinfo(1). It is *not* the track UID.

The special track ID '`-1`' is a wild card and applies the given switch to all tracks that are read from an input file.

The options that use the track IDs are the ones whose description contains '`TID`'. The following options use track IDs as well: `--atracks`, `--vtracks`, `--stracks` and `--btracks`.

Text files and character set conversions

Note

This section applies to all programs in MKVToolNix even if it only mentions mkvmerge(1).

All text in a Matroska™ file is encoded in UTF-8. This means that mkvmerge(1) has to convert every text file it reads as well as every text given on the command line from one character set into UTF-8. In return this also means thatmkvmerge(1)'s output has to be converted back to that character set from UTF-8, e.g. if a non-English translation is used with `--ui-language` or for text originating from a Matroska™ file.

mkvmerge(1) does this conversion automatically based on the presence of a *byte order marker* (short: BOM) or the system's current locale. How the character set is inferred from the locale depends on the operating system thatmkvmerge(1) is run on.

Text files that start with a BOM are already encoded in one representation of UTF. mkvmerge(1) supports the following five modes: UTF-8, UTF-16 Little and Big Endian, UTF-32 Little and Big Endian. Text files with a BOM are automatically

converted to UTF-8. Any of the parameters that would otherwise set the character set for such a file (e.g. `--sub-charset`) is silently ignored.

On Unix-like systems mkvmerge(1) uses the setlocale(3) system call which in turn uses the environment variables `LANG`, `LC_ALL` and `LC_CYPE`. The resulting character set is often one of UTF-8 or the ISO-8859-* family and is used for all text file operations and for encoding strings on the command line and for output to the console.

On Windows there are actually two different character sets that mkvmerge(1) uses due to the way the Windows shell program **cmd.exe** is implemented. The first character set is determined by a call to the `GetCP()` system call. This character set is used as the default for text file conversions and for all elements displayed by the GUI programs in the MKVToolNix package. **cmd.exe** uses another character set which is determined by a call to the `GetACP()` system call. This is the default character set for all strings read from the command line and for all strings output to the console.

The following options exist that allow specifying the character sets:

- `--sub-charset` for text subtitle files and for text subtitle tracks stored in container formats for which the character set cannot be determined unambiguously (e.g. Ogg files),
- `--chapter-charset` for chapter text files and for chapters and file titles stored in container formats for which the character set cannot be determined unambiguously (e.g. Ogg files for chapter information, track and file titles etc; MP4 files for chapter information),
- `--command-line-charset` for all strings on the command line,
- `--output-charset` for all strings written to the console or to a file if the output has been redirected with the `--redirect-output` option.

Subtitles

There are several text subtitle formats that can be embedded into Matroska™. At the moment mkvmerge(1) supports only text, VobSub and Kate subtitle formats. Text subtitles must be recoded to UTF-8 so that they can be displayed correctly by a player (see the section about text files and character sets for an explanation how mkvmerge(1) converts between character sets). Kate subtitles are already encoded in UTF-8 and do not have to be re-encoded.

The following subtitle formats are supported at the moment:

- Subtitle Ripper (SRT) files
- Substation Alpha (SSA) / Advanced Substation Alpha scripts (ASS)

- OggKate streams
- VobSub bitmap subtitle files

File linking

Matroska™ supports file linking which simply says that a specific file is the predecessor or successor of the current file. To be precise, it's not really the files that are linked but the Matroska™ segments. As most files will probably only contain one Matroska™ segment the following explanations use the term 'file linking' although 'segment linking' would be more appropriate.

Each segment is identified by a unique 128 bit wide segment UID. This UID is automatically generated by mkvmerge(1). The linking is done primarily via putting the segment UIDs (short: SID) of the previous/next file into the segment header information. mkvinfo(1) prints these SIDs if it finds them.

If a file is split into several smaller ones and linking is used then the timecodes will not start at 0 again but will continue where the last file has left off. This way the absolute time is kept even if the previous files are not available (e.g. when streaming). If no linking is used then the timecodes should start at 0 for each file. By default mkvmerge(1) does not use file linking. If you want that you can turn it on with the `--link` option. This option is only useful if splitting is activated as well.

Regardless of whether splitting is active or not the user can tell mkvmerge(1) to link the produced files to specific SIDs. This is achieved with the options `--link-to-previous` and `--link-to-next`. These options accept a segment SIDin the format that mkvinfo(1) outputs: 16 hexadecimal numbers between `0x00` and `0xff` prefixed with '`0x`' each, e.g. '`0x41 0xda 0x73 0x66 0xd9 0xcf 0xb2 0x1e 0xae 0x78 0xeb 0xb4 0x5e 0xca 0xb3 0x93`'. Alternatively a shorter form can be used: 16 hexadecimal numbers between `0x00` and `0xff` without the '`0x`' prefixes and without the spaces, e.g. '`41da7366d9cfb21eae78ebb45ecab393`'.

If splitting is used then the first file is linked to the SID given with `--link-to-previous` and the last file is linked to the SID given with `--link-to-next`. If splitting is not used then the one output file will be linked to both of the two SIDs.

Default values

The Matroska™ specification states that some elements have a default value. Usually an element is not written to the file if its value is equal to its default value in order to save space. The elements that the user might miss in mkvinfo(1)'s output are

the *language* and the *default track flag* elements. The default value for the *language* is English ('eng'), and the default value for the *default track flag* is *true*. Therefore if you used `--language 0:eng` for a track then it will not show up in mkvinfo(1)'s output.

Attachments

Maybe you also want to keep some photos along with your Matroska™ file, or you're using SSA subtitles and need a special TrueType™ font that's really rare. In these cases you can attach those files to the Matroska™ file. They will not be just appended to the file but embedded in it. A player can then show those files (the 'photos' case) or use them to render the subtitles (the 'TrueType™ fonts' case).

Here's an example how to attach a photo and a TrueType™ font to the output file:

```
$ mkvmerge -o output.mkv -A video.avi sound.ogg --attachment-description "Me
and the band behind the stage in a small get-together" --attachment-mime-type
image/jpeg --attach-file me_and_the_band.jpg --attachment-description "The
real rare and unbelievably good looking font" --attachment-type
application/octet-stream --attach-file really_cool_font.ttf
```

If a Matroska™ containing attachments file is used as an input file then mkvmerge(1) will copy the attachments into the new file. The selection which attachments are copied and which are not can be changed with the options `--attachments` and `--no-attachments`.

Chapters

The Matroska™ chapter system is more powerful than the old known system used by OGM files. The full specifications can be found at the Matroska™ website.

mkvmerge(1) supports two kinds of chapter files as its input. The first format, called '*simple chapter format*', is the same format that the OGM tools expect. The second format is a XML based chapter format which supports all ofMatroska™'s chapter functionality.

The simple chapter format

This formmat consists of pairs of lines that start with '`CHAPTERxx=`' and '`CHAPTERxxNAME=`' respectively. The first one contains the start timecode while the second one contains the title. Here's an example:

```
CHAPTER01=00:00:00.000
```

```
CHAPTER01NAME=Intro
CHAPTER02=00:02:30.000
CHAPTER02NAME=Baby prepares to rock
CHAPTER03=00:02:42.300
CHAPTER03NAME=Baby rocks the house
```

mkvmerge(1) will transform every pair or lines into one Matroska™ `ChapterAtom`. It does not set any `ChapterTrackNumber` which means that the chapters all apply to all tracks in the file.

As this is a text file character set conversion may need to be done. See the section about text files and character sets for an explanation how mkvmerge(1) converts between character sets.

The XML based chapter format

The XML based chapter format looks like this example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Chapters SYSTEM "matroskachapters.dtd">
<Chapters>
  <EditionEntry>
    <ChapterAtom>
      <ChapterTimeStart>00:00:30.000</ChapterTimeStart>
      <ChapterTimeEnd>00:01:20.000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapterString>A short chapter</ChapterString>
        <ChapterLanguage>eng</ChapterLanguage>
      </ChapterDisplay>
      <ChapterAtom>
        <ChapterTimeStart>00:00:46.000</ChapterTimeStart>
        <ChapterTimeEnd>00:01:10.000</ChapterTimeEnd>
        <ChapterDisplay>
          <ChapterString>A part of that short chapter</ChapterString>
          <ChapterLanguage>eng</ChapterLanguage>
        </ChapterDisplay>
      </ChapterAtom>
    </ChapterAtom>
  </EditionEntry>
</Chapters>
```

With this format three things are possible that are not possible with the simple chapter format:

1. The timestamp for the end of the chapter can be set,
2. chapters can be nested,
3. the language and country can be set.

The mkvtoolnix distribution contains some sample files in the `doc` subdirectory which can be used as a basis.

General notes

When splitting files mkvmerge(1) will correctly adjust the chapters as well. This means that each file only includes the chapter entries that apply to it, and that the timecodes will be offset to match the new timecodes of each output file.

mkvmerge(1) is able to copy chapters from Matroska™ source files unless this is explicitly disabled with the `--no-chapters` option. The chapters from all sources (Matroska™ files, Ogg files, MP4 files, chapter text files) are usually not merged but end up in separate `ChapterEditions`. Only if chapters are read from several Matroska™ or XML files that share the same edition UIDs will chapters be merged into a single `ChapterEdition`. If such a merge is desired in other situations as well then the user has to extract the chapters from all sources with mkvextract(1) first, merge the XML files manually and mux them afterwards.

Tags

Introduction

Matroska™ supports an extensive set of tags that is deprecated and a new, simpler system like it is is used in most other containers: *`KEY=VALUE`*. However, in Matroska™ these tags can also be nested, and both the *`KEY`* and the *`VALUE`* are elements of their own. The example file `example-tags-2.xml` shows how to use this new system.

Scope of the tags

Matroska™ tags do not automatically apply to the complete file. They can, but they also may apply to different parts of the file: to one or more tracks, to one or more chapters, or even to a combination of both. The the Matroska™ specification gives more details about this fact.

One important fact is that tags are linked to tracks or chapters with the `Targets` Matroska™ tag element, and that the UIDs used for this linking are *not* the track IDs mkvmerge(1) uses everywhere. Instead the numbers used are the UIDs which mkvmerge(1) calculates automatically (if the track is taken from a file format other than Matroska™) or which are copied from the source file if the track's source file is a Matroska™ file. Therefore it is difficult to know which UIDs to use in the tag file before the file is handed over to mkvmerge(1).

mkvmerge(1) knows two options with which you can add tags to Matroska™ files: The `--global-tags` and the `--tags` options. The difference is that the former option, `--global-tags`, will make the tags apply to the complete file by removing any of those `Targets` elements mentioned above. The latter option, `--tags`, automatically inserts the UID that mkvmerge(1) generates for the tag specified with the *`TID`* part of the `--tags` option.

Example

Let's say that you want to add tags to a video track read from an AVI. **mkvmerge --identify file.avi** tells you that the video track's ID (do not mix this ID with the UID!) is 0. So you create your tag file, leave out all `Targets` elements and call mkvmerge(1):

```
$ mkvmerge -o file.mkv --tags 0:tags.xml file.avi
```

Tag file format

mkvmerge(1) supports a XML based tag file format. The format is very closely modeled after the Matroska™ specification. Both the binary and the source distributions of MKVToolNix come with a sample file called `example-tags-2.xml` which simply lists all known tags and which can be used as a basis for real life tag files.

The basics are:

- The outermost element must be `<Tags>`.
- One logical tag is contained inside one pair of `<Tag>` XML tags.
- White spaces directly before and after tag contents are ignored.

Data types

The new Matroska™ tagging system only knows two data types, a UTF-8 string and a binary type. The first is used for the tag's name and the `<String>` element while the binary type is used for the `<Binary>` element.

As binary data itself would not fit into a XML file mkvmerge(1) supports two other methods of storing binary data. If the contents of a XML tag starts with '`@`' then the following text is treated as a file name. The corresponding file's content is copied into the Matroska™ element.

Otherwise the data is expected to be *Base64* encoded. This is an encoding that transforms binary data into a limited set of ASCII characters and is used e.g. in email programs. mkvextract(1) will output *Base64* encoded data for binary elements.

The deprecated tagging system knows some more data types which can be found in the official Matroska™ tag specs. As mkvmerge(1) does not support this system anymore these types aren't described here.

Matroska™ file layout

The Matroska™ file layout is quite flexible. mkvmerge(1) will render a file in a predefined way. The resulting file looks like this:

[EBML head] [segment {meta seek #1} [segment information] [track information] {attachments} {chapters} [cluster 1] {cluster 2} ... {cluster n} {cues} {meta seek #2} {tags}]

The elements in curly braces are optional and depend on the contents and options used. A couple of notes:

- meta seek #1 includes only a small number of level 1 elements, and only if they actually exist: attachments, chapters, cues, tags, meta seek #2. Older versions of mkvmerge(1) used to put the clusters into this meta seek element as well. Therefore some imprecise guessing was necessary to reserve enough space. It often failed. Now only the clusters are stored in meta seek #2, and meta seek #1 refers to the meta seek element #2.
- Attachment, chapter and tag elements are only present if they were added.

The shortest possible Matroska file would look like this:

[EBML head] [segment [segment information] [track information] [cluster 1]]

This might be the case for audio-only files.

External timecode files

mkvmerge(1) allows the user to chose the timecodes for a specific track himself. This can be used in order to create files with variable frame rate video or include gaps in audio. A frame in this case is the unit that mkvmerge(1) creates separately per Matroska™ block. For video this is exactly one frame, for audio this is one packet of the specific audio type. E.g. for AC3 this would be a packet containing `1536` samples.

Timecode files that are used when tracks are appended to each other must only be specified for the first part in a chain of tracks. For example if you append two files, v1.avi and v2.avi, and want to use timecodes then your command line must look something like this:

```
mkvmerge ... --timecodes 0:my_timecodes.txt v1.avi +v2.avi
```

There are four formats that are recognized by mkvmerge(1). The first line always contains the version number. Empty lines, lines containing only whitespace and lines beginning with '`#`' are ignored.

Timecode file format v1

This format starts with the version line. The second line declares the default number of frames per second. All following lines contain three numbers separated by commas: the start frame (`0` is the first frame), the end frame and the number of frames in this range. The FPS is a floating point number with the dot '.' as the decimal point. The ranges can contain gaps for which the default FPS is used. An example:

```
# timecode format v1
assume 27.930
800,1000,25
1500,1700,30
```

Timecode file format v2

In this format each line contains a timecode for the corresponding frame. This timecode must be given in millisecond precision. It can be a floating point number, but it doesn't have to be. You *have to* give at least as many timecode lines as there are frames in the track. The timecodes in this file must be sorted. Example for 25fps:

```
# timecode format v2
0
40
80
```

Timecode file format v3

In this format each line contains a duration in seconds followed by an optional number of frames per second. Both can be floating point numbers. If the number of frames per second is not present the default one is used. For audio you should let the codec calculate the frame timecodes itself. For that you should be using `0.0` as the number of

frames per second. You can also create gaps in the stream by using the '`gap`' keyword followed by the duration of the gap. Example for an audio file:

```
# timecode format v3
assume 0.0
25.325
7.530,38.236
gap, 10.050
2.000,38.236
```

Timecode file format v4

This format is identical to the v2 format. The only difference is that the timecodes do not have to be sorted. This format should almost never be used.

Exit codes

mkvmerge(1) exits with one of three exit codes:

- `0` -- This exit codes means that muxing has completed successfully.
- `1` -- In this case mkvmerge(1) has output at least one warning, but muxing did continue. A warning is prefixed with the text '`Warning:`'. Depending on the issues involved the resulting file might be ok or not. The user is urged to check both the warning and the resulting file.
- `2` -- This exit code is used after an error occurred. mkvmerge(1) aborts right after outputting the error message. Error messages range from wrong command line arguments over read/write errors to broken files.

See also

mkvinfo(1), mkvextract(1), mkvpropedit(1), mmg(1)

WWW

The latest version can always be found at the MKVToolNix homepage.