mkvmerge -- Merge multimedia streams into a *Matroska*(tm) file

Table of contents

1. Synopsis

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

2. Description

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

> **Important:**
>
> The order of command line options is important. Please read the section "Option order" if you're new to the program.

2.1. Global options

Option	Description
`-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, VP9 video and Opus, Vorbis audio tracks. The DocType header item is changed to "webm". For chapters and tags only a subset of elements are allowed. *mkvmerge*(1) will automatically remove all elements not allowed by the specification.
`--title` *title*	Sets the general title for the output file, e.g. the movie name.
`--default-language` *language-code*	Sets the default language code that will be used for tracks for which no language is set with the --language option and for which the source container doesn't provide a language. The default language code is '`und`' for 'undefined'.

2.2. Segment info handling (global options)

Option	Description
`--segmentinfo` *filename.xml*	Read segment information from an 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. See the section about segment info XML files below for details.

Option	Description
`--segment-uid SID1,SID2,...`	Sets the segment UIDs to use. This is a comma-separated list of 128-bit segment UIDs in the usual UID form: hex numbers with or without the "0x" prefix, with or without spaces, exactly 32 digits. If SID starts with = then its rest is interpreted as the name of a Matroska file whose segment UID is read and used. 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.

2.3. Chapter and tag handling (global options)

Option	Description
`--chapter-language Language-code`	Sets the ISO 639-2 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. The language set with this option is also used when chapters are generated with the --generate-chapters option.
`--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.
`--chapter-sync d[,o[/p]]`	Adjust the timestamps of the chapters in the following source file by *d* ms. Alternatively you can use the `--sync` option with the special track ID `-2` (see section special 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.
`--generate-chapters mode`	*mkvmerge*(1) can create chapters automatically. The following two modes are currently supported: • '`when-appending`' – This mode creates one chapter at the start and one chapter whenever a file is appended. This mode also works with split modes '`parts:`' and '`parts-frames:`'. For these modes one chapter will be generated for each appended timestamp range (those whose start timestamps are prefixed with '`+`'). **Note:** *mkvmerge*(1) requires a video or an audio track to be present in order to be able to determine when a new file is appended. If one or more video tracks are muxed the first one is used. Otherwise the first audio track is used. • '`interval:time-spec`' – This mode creates one chapter at fixed intervals given by *time-spec*. The format is either the form *HH:MM:SS.nnnnnnnnn* or a number followed by one of the units '`s`', '`ms`' or '`us`'. Example: `--generate-chapters interval:45s` The names for the new chapters are controlled by the option --generate-chapters-name-template. The language is set with --chapter-language which must occur before `--generate-chapters`.

Option	Description
--generate-chapters-name-template *template*	This sets the name template for chapter names generated by the option --generate-chapters. If the option is not used then default '<NUM:2>' will be used. There are several variables that can be used in the template that are replaced by their actual values when a chapter is generated. The string '<NUM>' will be replaced by the chapter number. The string '<START>' will be replaced by the chapter's start timestamp. The strings '<FILE_NAME>' and '<FILE_NAME_WITH_EXT>' are only filled when generating chapters for appended files. They will be replaced by the appended file's name wihtout respectively with its extension. Note that only the file's base name and extension are inserted, not its directory or drive components. You can specify a minimum number of places for the chapter number with '<NUM:places>', e.g. '<NUM:3>'. The resulting number will be padded with leading zeroes if the number of places is less than specified. You can control the format used by the start timestamp with <START:format>. The format defaults to '%H:%M:%S' if none is given. Valid format codes are: • %h – hours • %H – hours zero-padded to two places • %m – minutes • %M – minutes zero-padded to two places • %s – seconds • %S – seconds zero-padded to two places • %n – nanoseconds with nine places • %<1-9>n – nanoseconds with up to nine places (e.g. three places with %3n)
--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.

2.4. General output control (advanced global options)

Option	Description
--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.
--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.

Option	Description
`--timestamp-scale` *factor*	Forces the timestamp 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 timestamps 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 timestamp scale factor so that all timestamps 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.
`--enable-durations`	Write durations for all blocks. This will increase file size and does not offer any additional value for players at the moment.
`--no-cues`	Tells *mkvmerge*(1) not to create and write the cue data which can be compared to an index in an AVI. *Matroska*(tm) 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.
`--no-date`	By default *mkvmerge*(1) sets the "date" segment information field to the time & date when multiplexing started. With this option that field is not written at all.
`--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.
`--disable-track-statistics-tags`	Normally *mkvmerge*(1) will write certain tags with statistics for each track. If such tags are already present then they will be overwritten. The tags are `BPS`, `DURATION`, `NUMBER_OF_BYTES` and `NUMBER_OF_FRAMES`. Enabling this option prevents *mkvmerge*(1) from writing those tags and from touching any existing tags with same names.
`--disable-language-ietf`	Normally *mkvmerge*(1) will write the new IETF BCP 47 language elements in addition to the legacy language elements in track headers, chapters and tags. If this option is used, only the legacy elements are written.

2.5. File splitting, linking, appending and concatenation (more global options)

Option	Description
`--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. Therefore the split point may be a bit off from what the user has specified. At the moment *mkvmerge*(1) supports the following 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 timestamps. Syntax: `--split timestamps:`*A*[,*B*[,*C*...]] Example: `--split timestamps: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 timestamps is separated by commas. After the input stream has reached the current split point's timestamp a new file is created. Then the next split point given in this list is used. The '`timestamps:`' prefix must not be omitted. 4. Keeping specific parts by specifying timestamp ranges while discarding others.

Option	Description

Syntax: `--split parts:start1-end1[,[+]start2-end2[,[+]start3-end3...]]`

Examples:

1. `--split parts:00:01:20-00:02:45,00:05:50-00:10:30`
2. `--split parts:00:01:20-00:02:45,+00:05:50-00:10:30`
3. `--split parts:-00:02:45,00:05:50-`

The `parts` mode tells *mkvmerge*(1) to keep certain ranges of timestamps while discarding others. The ranges to keep have to be listed after the `parts:` keyword and be separated by commas. A range itself consists of a start and an end timestamp in the same format the other variations of `--split` accept (e.g. both `00:01:20` and `80s` refer to the same timestamp).

If a start timestamp is left out then it defaults to the previous range's end timestamp. If there was no previous range then it defaults to the start of the file (see example 3).

If an end timestamp is left out then it defaults to the end of the source files which basically tells *mkvmerge*(1) to keep the rest (see example 3).

Normally each range will be written to a new file. This can be changed so that consecutive ranges are written to the same file. For that the user has to prefix the start timestamp with a `+`. This tells *mkvmerge*(1) not to create a new file and instead append the range to the same file the previous range was written to. Timestamps will be adjusted so that there will be no gap in the output file even if there was a gap in the two ranges in the input file.

In example 1 *mkvmerge*(1) will create two files. The first will contain the content starting from `00:01:20` until `00:02:45`. The second file will contain the content starting from `00:05:50` until `00:10:30`.

In example 2 *mkvmerge*(1) will create only one file. This file will contain both the content starting from `00:01:20` until `00:02:45` and the content starting from `00:05:50` until `00:10:30`.

In example 3 *mkvmerge*(1) will create two files. The first will contain the content from the start of the source files until `00:02:45`. The second file will contain the content starting from `00:05:50` until the end of the source files.

> **Note:**
>
> Note that *mkvmerge*(1) only makes decisions about splitting at key frame positions. This applies to both the start and the end of each range. So even if an end timestamp is between two key frames *mkvmerge*(1) will continue outputting the frames up to but excluding the following key frame.

5. Keeping specific parts by specifying frame/field number ranges while discarding others.

Syntax: `--split parts-frames:start1-end1[,[+]start2-end2[,[+]start3-end3...]]`

Examples:

1. `--split parts-frames:137-258,548-1211`
2. `--split parts-frames:733-912,+1592-2730`
3. `--split parts-frames:-430,2512-`

The `parts-frames` mode tells *mkvmerge*(1) to keep certain ranges of frame/field numbers while discarding others. The ranges to keep have to be listed after the `parts-frames:` keyword and be separated by commas. A range itself consists of a start and an end frame/field number. Numbering starts at 1.

If a start number is left out then it defaults to the previous range's end number. If there was no previous range then it defaults to the start of the file (see example 3).

If an end number is left out then it defaults to the end of the source files which basically tells *mkvmerge*(1) to keep the rest (see example 3).

Normally each range will be written to a new file. This can be changed so that consecutive ranges are written to the same file. For that the user has to prefix the start number with a `+`. This tells *mkvmerge*(1) not to create a new file and instead append the range to the same file the previous range was written to. Timestamps will be adjusted so that there will be no gap in the output file even if there was a gap in the two ranges in the input file.

> **Note:**
>
> Note that *mkvmerge*(1) only makes decisions about splitting at key frame positions. This applies to both the start and the end of each range. So even if an end frame/field number is between two key

Option	Description
	frames *mkvmerge*(1) will continue outputting the frames up to but excluding the following key frame. In example 1 *mkvmerge*(1) will create two files. The first will contain the content starting from the first key frame at or after `137` up to but excluding the first key frame at or after `258`. The second file will contain the content starting from `548` until `1211`. In example 2 *mkvmerge*(1) will create only one file. This file will contain both the content starting from `733` until `912` and the content starting from `1592` until `2730`. In example 3 *mkvmerge*(1) will create two files. The first will contain the content from the start of the source files until `430`. The second file will contain the content starting from `2512` until the end of the source files. This mode considers only the first video track that is output. If no video track is output no splitting will occur. **Note:** The numbers given with this argument are interpreted based on the number of *Matroska*(tm) blocks that are output. A single *Matroska*(tm) block contains either a full frame (for progressive content) or a single field (for interlaced content). mkvmerge does not distinguish between those two and simply counts the number of blocks. For example: If one wanted to split after the 25th full frame with interlaced content one would have to use `50` (two fields per full frame) as the split point. 6. Splitting after specific frames/fields. Syntax: `--split frames:`*A*[,*B*[,*C*...]] Example: `--split frames:120,237,891` The parameters *A*, *B*, *C* etc must all be positive integers. Numbering starts at 1. The list of frame/field numbers is separated by commas. After the input stream has reached the current split point's frame/field number a new file is created. Then the next split point given in this list is used. The '`frames:`' prefix must not be omitted. This mode considers only the first video track that is output. If no video track is output no splitting will occur. **Note:** The numbers given with this argument are interpreted based on the number of *Matroska*(tm) blocks that are output. A single *Matroska*(tm) block contains either a full frame (for progressive content) or a single field (for interlaced content). mkvmerge does not distinguish between those two and simply counts the number of blocks. For example: If one wanted to split after the 25th full frame with interlaced content one would have to use `50` (two fields per full frame) as the split point. 7. Splitting before specific chapters. Syntax: `--split chapters:all` or `--split chapters:`*A*[,*B*[,*C*...]] Example: `--split chapters:5,8` The parameters *A*, *B*, *C* etc must all be positive integers. Numbering starts at 1. The list of chapter numbers is separated by commas. Splitting will occur right before the first key frame whose timestamp is equal to or bigger than the start timestamp for the chapters whose numbers are listed. A chapter starting at 0s is never considered for splitting and discarded silently. The keyword `all` can be used instead of listing all chapter numbers manually. The '`chapters:`' prefix must not be omitted. **Note:** The *Matroska*(tm) file format supports arbitrary deeply nested chapter structures called 'edition entries' and 'chapter atoms'. However, this mode only considers the top-most level of chapters across all edition entries. 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.

Option	Description
	Another possible pattern is '%c' which will be replaced by the name of the first chapter in the file. Note that when '%c' is present, the pattern '-%03d' will not be added automatically.
--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. If SID starts with = then its rest is interpreted as the name of a Matroska file whose segment UID is read and used.
--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. If SID starts with = then its rest is interpreted as the name of a Matroska file whose segment UID is read and used.
--append-mode *mode*	Determines how timestamps 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 timestamps for 'track2_1' by an amount. For 'file' mode this amount is the highest timestamp encountered in 'file1' even if that timestamp was from a different track than 'track1_1'. In track mode the offset is the highest timestamp 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 or MP4 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`

Option	Description

[*file1* *file2*]	If multiple file names are contained in a pair of square brackets then the second and all following files will be appended to the first file named within the brackets.
	
	This is an alternative syntax to using '+' between the file names. Therefore the following two commands are equivalent:
	
	`$ mkvmerge -o full.mkv file1.mkv + file2.mkv`
	`$ mkvmerge -o full.mkv '[' file1.mkv file2.mkv ']'`

=	For certain file types (MPEG program streams = VOBs) *mkvmerge*(1) normally 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) and treats all of those files as if they were concatenated into a single big file. 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.vob`
	`$ mkvmerge -o full.mkv =file1.vob`

(*file1* *file2*)	If multiple file names are contained in a pair of parenthesis then those files will be treated as if they were concatenated into a single big file consisting of the content of each of the files one after the other.
	
	This can be used for e.g. VOB files coming from a DVD or MPEG transport streams. It cannot be used if each file contains its own set of headers which is usually the case with stand-alone files like AVI or MP4.
	
	Putting a file name into parenthesis also prevents *mkvmerge*(1) from looking for additional files with the same base name as described in option =. Therefore these two command lines are equivalent:
	
	`$ mkvmerge -o out.mkv = file.mkv`
	`$ mkvmerge -o out.mkv '(' file.mkv ')'`
	
	Several things should be noted:
	
	1. There must be spaces both after the opening and before the closing parenthesis.
	2. Every parameter between parenthesis is interpreted as a file name. Therefore all options applying to this logical file must be listed before the opening parenthesis.
	3. Some shells treat parenthesis as special characters. Hence you must escape or quote them as shown in the example above.

2.6. Attachment support (more global options)

Option	Description
`--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*(tm) 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*(tm) file.

2.7. Options that can be used for each input file

Option	Description
`-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. Instead of track IDs you can also provide ISO 639-2 language codes. This will only work for source files that provide language tags for their tracks. Default: copy all tracks of this kind. If the IDs are prefixed with `!` then the meaning is reversed: copy all tracks of this kind but the ones listed after the `!`.
`-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. Instead of track IDs you can also provide ISO 639-2 language codes. This will only work for source files that provide language tags for their tracks. If the IDs are prefixed with `!` then the meaning is reversed: copy all tracks of this kind but the ones listed after the `!`.
`-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. Instead of track IDs you can also provide ISO 639-2 language codes. This will only work for source files that provide language tags for their tracks. If the IDs are prefixed with `!` then the meaning is reversed: copy all tracks of this kind but the ones listed after the `!`.
`-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. Instead of track IDs you can also provide ISO 639-2 language codes. This will only work for source files that provide language tags for their tracks. If the IDs are prefixed with `!` then the meaning is reversed: copy all tracks of this kind but the ones listed after the `!`.
`--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. If the IDs are prefixed with `!` then the meaning is reversed: copy everything but the IDs listed after the `!`.

Option	Description
`-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. If the IDs are prefixed with `!` then the meaning is reversed: copy everything but the IDs listed after the `!`.
`-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.
`-y`, `--sync` *TID:d[,o[/p]]*	Adjust the timestamps 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 and subtitle tracks and '`none`' for audio tracks. 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 track" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. The flag will be set if the source container doesn't provide that information and the user doesn't specify it via this option. If the user does not explicitly select a track during playback, the player should select one of the tracks that has its "default track" flag set, taking user preferences such as their preferred language into account. 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 display" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. Use this for tracks containing onscreen text or foreign-language dialogue. This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.
`--hearing-impaired-flag` *TID[:bool]*	Sets the "hearing impaired" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. This flag can be set if the track is suitable for users with hearing impairments. This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.
`--visual-impaired-flag` *TID[:bool]*	Sets the "visual impaired" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. This flag can be set if the track is suitable for users with visual impairments. This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.

Option	Description
`--text-descriptions-flag` *TID[:bool]*	Sets the "text descriptions" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. This flag can be set if the track contains textual descriptions of video content suitable for playback via a text-to-speech system for a visually-impaired user. This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.
`--original-flag` *TID[:bool]*	Sets the "original language" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. This flag can be set if the track is in the content's original language (not a translation). This option can be used multiple times for an input file applying to several tracks by selecting different track IDs each time.
`--commentary-flag` *TID[:bool]*	Sets the "commentary" flag for the given track (see section track IDs) if the optional argument *bool* is set to `1` or if it isn't present. This flag can be set if the track contains commentary. 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 ISO 639-2 language codes and ISO 639-1 country codes are allowed. The country codes will be converted to language codes automatically. All languages including their ISO 639-2 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*(tm) file) and b) the AAC file contains SBR AAC data. 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 decoders *mkvmerge*(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*(tm) 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.
`--reduce-to-core` *TID*	Some audio codecs have a lossy core and optional extensions that implement lossless decoding. This option tells *mkvmerge*(1) to only copy the core but not the extensions. By default *mkvmerge*(1) copies both the core and the extensions. Currently only DTS tracks are affected by this option. TrueHD tracks that contain an embedded AC-3 core are instead presented as two separate tracks for which the user can select which track to copy. For DTS such a scheme would not work as the HD extensions cannot be decoded by themselves – unlike the TrueHD data.
`--remove-dialog-normalization-gain` *TID*	Some audio codecs contain header fields that tell the decoder or player to apply a (usually negative) gain for dialog normalization. This option tells *mkvmerge*(1) to remove or minimize that gain by modifying the corresponding header fields. Currently only AC-3, DTS and TrueHD tracks are affected by this option.
`--timestamps` *TID:file-name*	Read the timestamps to be used for the specific track ID from *file-name*. These timestamps forcefully override the timestamps that *mkvmerge*(1) normally calculates. Read the section about external timestamp files.
	

Option	Description
`--default-duration` *TID:x*	Forces the default duration of a given track to the specified value. Also modifies the track's timestamps to match the default duration. The argument *x* must be postfixed with 's', 'ms', 'us', 'ns', 'fps', 'p' or 'i' to specify the default duration in seconds, milliseconds, microseconds, nanoseconds, 'frames per second', 'progressive frames per second' or 'interlaced 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 the encoded bitstream for certain track types, e.g. AVC/H.264 or MPEG-2. This option can also be used to change the FPS of video tracks without having to use an external timestamp file.
`--fix-bitstream-timing-information` *TID[:0|1]*	Normally *mkvmerge*(1) does not change the timing information (frame/field rate) stored in the video bitstream. With this option that information is adjusted to match the container timing information. The container timing information can come from various sources: from the command line (see option --default-duration), the source container or derived from the bitstream. **Note:** This has only been implemented for AVC/H.264 video tracks so far.
`--compression` *TID:n*	Selects the compression method to be used for the track. Note that the player also has to support this method. Valid values are 'none', 'zlib' and 'mpeg4_p2'/'mpeg4p2'. 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 default for some subtitle types 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.

2.8. Options that only apply to video tracks

Option	Description
`-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*(tm) 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*(tm) 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.

Option	Description
`--colour-matrix-coefficients` *TID:n*	Sets the matrix coefficients of the video used to derive luma and chroma values from red, green and blue color primaries. The parameter *n* is an integer rangeing from `0` and `10`. Valid values and their meaning are: `0`: GBR, `1`: BT709, `2`: unspecified, `3`: reserved, `4`: FCC, `5`: BT470BG, `6`: SMPTE 170M, `7`: SMPTE 240M, `8`: YCOCG, `9`: BT2020 non-constant luminance, `10`: BT2020 constant luminance
`--colour-bits-per-channel` *TID:n*	Sets the number of coded bits for a colour channel. A value of `0` indicates that the number of bits is unspecified.
`--chroma-subsample` *TID:hori,vert*	The amount of pixels to remove in the Cr and Cb channels for every pixel not removed horizontally/vertically. Example: For video with 4:2:0 chroma subsampling, the parameter should be set to *TID*`:1,1`.
`--cb-subsample` *TID:hori,vert*	The amount of pixels to remove in the Cb channel for every pixel not removed horizontally/vertically. This is additive with `--chroma-subsample`. Example: For video with 4:2:1 chroma subsampling, the parameter `--chroma-subsample` should be set to *TID*`:1,0` and Cb-subsample should be set to *TID*`:1,0`.
`--chroma-siting` *TID:hori,vert*	Sets how chroma is sited horizontally/vertically (`0`: unspecified, `1`: top collocated, `2`: half).
`--colour-range` *TID:n*	Sets the clipping of the color ranges (`0`: unspecified, `1`: broadcast range, `2`: full range (no clipping), `3`: defined by MatrixCoefficients/TransferCharacteristics).
`--colour-transfer-characteristics` *TID:n*	The transfer characteristics of the video. Valid values and their meaning are: `0`: reserved, `1`: ITU-R BT.709, `2`: unspecified, `3`: reserved, `4`: gamma 2.2 curve, `5`: gamma 2.8 curve, `6`: SMPTE 170M, `7`: SMPTE 240M, `8`: linear, `9`: log, `10`: log sqrt, `11`: IEC 61966-2-4, `12`: ITU-R BT.1361 extended colour gamut, `13`: IEC 61966-2-1, `14`: ITU-R BT.2020 10 bit, `15`: ITU-R BT.2020 12 bit, `16`: SMPTE ST 2084, `17`: SMPTE ST 428-1; `18`: ARIB STD-B67 (HLG)
`--colour-primaries` *TID:n*	Sets the colour primaries of the video. Valid values and their meaning are: `0`: reserved, `1`: ITU-R BT.709, `2`: unspecified, `3`: reserved, `4`: ITU-R BT.470M, `5`: ITU-R BT.470BG, `6`: SMPTE 170M, `7`: SMPTE 240M, `8`: FILM, `9`: ITU-R BT.2020, `10`: SMPTE ST 428-1, `22`: JEDEC P22 phosphors
`--max-content-light` *TID:n*	Sets the maximum brightness of a single pixel (Maximum Content Light Level) in candelas per square meter (cd/m²). The value of *n* should be a non-negtive integer.
`--max-frame-light` *TID:n*	Sets the maximum brightness of a single full frame (Maximum Frame-Average Light Level) in candelas per square meter (cd/m²). The value of *n* should be a non-negtive integer.
`--chromaticity-coordinates` *TID:red-x,red-y,green-x,green-y,blue-x,blue-y*	Sets the red/green/blue chromaticity coordinates as defined by CIE 1931.
`--white-colour-coordinates` *TID:x,y*	Sets the white colour chromaticity coordinates as defined by CIE 1931.
`--max-luminance` *TID:float*	Sets the maximum luminance in candelas per square meter (cd/m²). The value should be less than 9999.99.
`--min-luminance` *TID:float*	Sets the minimum luminance in candelas per square meter (cd/m²). The value should be less than 999.9999.
`--projection-type` *TID:method*	Sets the video projection method used. Valid values are 0 (rectangular projection), 1 (equirectangular projection), 2 (cubemap projection) and 3 (mesh projection).
`--projection-private` *TID:data*	Sets private data that only applies to a specific projection. Data must be given as hex numbers with or without the "0x" prefix, with or without spaces.
`--projection-pose-yaw` *TID:float*	Specifies a yaw rotation to the projection.
`--projection-pose-pitch` *TID:float*	Specifies a pitch rotation to the projection.
`--projection-pose-roll` *TID:float*	Specifies a roll rotation to the projection.

Option	Description
`--field-order` *TID:n*	Sets the field order for the video track with the track ID *TID*. The order must be one of the following numbers: `0`: progressive; `1`: interlaced with top field displayed first and top field stored first; `2`: undetermined field order; `6`: interlaced with bottom field displayed first and bottom field stored first; `9`: interlaced with bottom field displayed first and top field stored first; `14`: interlaced with top field displayed first and bottom field stored first
`--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 `14` or one of these keywords: `'mono'`, `'side_by_side_left_first'`, `'top_bottom_right_first'`, `'top_bottom_left_first'`, `'checkerboard_right_first'`, `'checkerboard_left_first'`, `'row_interleaved_right_first'`, `'row_interleaved_left_first'`, `'column_interleaved_right_first'`, `'column_interleaved_left_first'`, `'anaglyph_cyan_red'`, `'side_by_side_right_first'`, `'anaglyph_green_magenta'`, `'both_eyes_laced_left_first'`, `'both_eyes_laced_right_first'`.

2.9. Options that only apply to text subtitle tracks

Option	Description
`--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 from *Matroska*(tm) 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.

2.10. Other options

Option	Description
`-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. The output format used for the result can be changed with the option --identification-format.
`-J` *file-name*	This is a convenient alias for "`--identification-format json --identify file-name`".
`-F`, `--identification-format` *format*	Determines the output format used by the --identify option. The following formats are supported: `text` (the default if this option isn't used) and `json`. 1. The `text` format is short and human-readable. It consists of one line per item found (container, tracks, attachments etc.). This format is not meant to be parsed. The output will be translated into the language *mkvmerge*(1) uses (see also --ui-language). 2. The `json` format outputs a machine-readable JSON representation. This format follows the JSON schema described in the following file: mkvmerge-identification-output-schema-v14.json All versions of the JSON schema are available both online and in the released source code archives.
`--probe-range-percentage` *percentage*	File types such as MPEG program and transport streams (`.vob`, `.m2ts`) require parsing a certain amount of data in order to detect all tracks contained in the file. This amount is 0.3% of the source file's size or 10 MB, whichever is higher. If tracks are known to be present but not found then the percentage to probe can be changed with this option. The minimum of 10 MB is built-in and cannot be changed.
`-l`, `--list-types`	Lists supported input file types.
`--list-languages`	Lists all languages and their ISO 639-2 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. Selecting `'lowest'` also causes *mkvmerge*(1) to select idle I/O priority in addition to the lowest possible process priority.
`--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.

Option	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.
`--flush-on-close`	Tells the program to flush all data cached in memory to storage when closing files opened for writing. This can be used to prevent data loss on power outages or to circumvent certain problems in the operating system or drivers. The downside is that multiplexing will take longer as mkvmerge will wait until all data has been written to the storage before exiting. See issues #2469 and #2480 on the MKVToolNix bug tracker for in-depth discussions on the pros and cons.
`--ui-language` *code*	Forces the translations for the language *code* to be used (e.g. '`de_DE`' for the German translations). Entering '`list`' as the *code* will cause the program to output a list of available translations.
`--abort-on-warnings`	Tells the program to abort after the first warning is emitted. The program's exit code will be 1.
`--deterministic` *seed*	Enables the creation of byte-identical files if the same version of *mkvmerge*(1) is used with the same source files, the same set of options and the same seed. Note that the "date" segment information field is not written in this mode. The seed can be an arbitrary string and does not have to be a number. The result of byte-identical files is only guaranteed under the following conditions: 1. The same version of *mkvmerge*(1) built with the same versions of libEBML and libMatroska is used. 2. The source files used are byte-identical. 3. The same command line options are used in the same order (with the notable exception of *--output …*). Using other versions of *mkvmerge*(1) or other command-line options may result in the same byte-identical file but is not guaranteed to do so.
`--debug` *topic*	Turn on debugging for a specific feature. This option is only useful for developers.
`--engage` *feature*	Turn on experimental features. A list of available features can be requested with `mkvmerge --engage list`. These features are not meant to be used in normal situations.
`--gui-mode`	Turns on GUI mode. In this mode specially-formatted lines may be output that can tell a controlling GUI what's happening. These messages follow the format '`#GUI#message`'. The message may be followed by key/value pairs as in '`#GUI#message#key1=value1#key2=value2…`'. Neither the messages nor the keys are ever translated and always output in English.
`@`*options-file.json*	Reads additional command line arguments from the file *options-file*. See the section about option files for further information.
`--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: • '`FLAC`' -- reading raw FLAC files and handling FLAC tracks in other containers, e.g. *Ogg*(tm) or *Matroska*(tm).
`-h`, `--help`	Show usage information and exit.
`-V`, `--version`	Show version information and exit.

3. 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.

> **Important:**
>
> The order of command line options is important. Please read the section "Option order" if you're new to the program.

4. Option order

The order in which options are entered is important for some options. Options fall into two categories:

1. Options that affect the whole program and are not tied to any input file. These include but are not limited to `--command-line-charset`, `--output` or `--title`. These can appear anywhere on the command line.
2. Options that affect a single input file or a single track in an input file. These options all apply to the following input file on the command line. All options applying to the same input (or to tracks from the same input file) file can be written in any order as long as they all appear before that input file's name. Examples for options applying to an input file are `--no-chapters` or `--chapter-charset`. Examples for options applying to a single track are `--default-duration` or `--language`.

The options are processed from left to right. If an option appears multiple times within the same scope then the last occurrence will be used. Therefore the title will be set to "Something else" in the following example:

```
$ mkvmerge -o output.mkv --title 'This and that' input.avi --title 'Something else'
```

The following example shows that using the `--language` option twice is OK because they're used in different scopes. Even though they apply to the same track ID they apply to different input files and therefore have different scopes:

```
$ mkvmerge -o output.mkv --language 0:fre français.ogg --language 0:deu deutsch.ogg
```

5. 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*(tm):

```
$ 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*(tm) and enjoy. If you have multiple audio tracks (or even video tracks) then you can tell *mplayer*(tm) 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*(tm)) or the *subrip*(tm) 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 ISO 639-2 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*(tm) 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 --defa
```

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.

Turn off the compression for an input file.

```
$ mkvmerge -o no-compression.mkv --compression -1:none MyMovie.avi --compression -1:none mymovie.srt
```

6. Track IDs

6.1. Regular 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 0: video (V_MS/VFW/FOURCC, DIV3)
Track ID 1: audio (A_MPEG/L3)
```

Do not confuse the track IDs that are assigned to the tracks that are placed in the output MKV file with the track IDs of the input files. Only the input file track IDs are used for options needing these values.

Also note that each input file has its own set of track IDs. Therefore the track IDs for file '`file1.ext`' as reported by '`mkvmerge --identify`' do not change no matter how many other input files are there or in which position '`file1.ext`' is used.

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, AC-3, MP3, SRT and WAV files: The one 'track' in that file gets the ID 0.
- Most other files: The track IDs are assigned in order the tracks are found in the file starting at 0.

The options that use the track IDs are the ones whose description contains '`TID`'. The following options use track IDs as well: `--audio-tracks`, `--video-tracks`, `--subtitle-tracks`, `--button-tracks` and `--track-tags`.

6.2. Special track IDs

There are several IDs that have special meaning and do not occur in the identification output.

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 special track ID '`-2`' refers to the chapters in a source file. Currently only the `--sync` option uses this special ID. As an alternative to `--sync -2:…` the option `--chapter-sync …` can be used.

7. Text files and character set conversions

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

7.1. Introduction

All text in a *Matroska*(tm) 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 that *mkvmerge*(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*(tm) 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 that *mkvmerge*(1) is run on.

7.2. Byte order markers (BOM)

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.

7.3. Linux and Unix-like systems including macOS

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.

7.4. Windows

On Windows the default character set used for converting text files is determined by a call to the `GetACP()` system call.

Reading the command line is done with the `GetCommandLineW()` function which already returns a Unicode string. Therefore the option `--command-line-charset` is ignored on Windows.

Output to the console consists of three scenarios:

1. If the output is redirected with the option --redirect-output then the default charset is UTF-8. This can be changed with --output-charset.
2. If the output is redirected with `cmd.exe` itself, e.g. with `mkvinfo file.mkv > info.txt`, then the charset is always UTF-8 and cannot be changed.
3. Otherwise (when writing directly to the console) the Windows function `WriteConsoleW()` is used and the option --output-charset is ignored. The console should be able to output all Unicode characters for which the corresponding language support is installed (e.g. Chinese characters might not be displayed on English Windows versions).

7.5. Command line options

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. On non-Windows systems the default for the output charset is the system's current charset. On Windows it defaults to UTF-8 both for redirecting with --redirect-output and with `cmd.exe` itself, e.g. `mkvinfo file.mkv > info.txt`.

8. Option files

An option file is a file *mkvmerge*(1) can read additional command line arguments from. This can be used in order to circumvent certain limitations of the shell or the operating system when executing external programs like a limited command line length.

An option file contains JSON-formatted data. Its content must be a valid JSON array consisting solely of JSON strings. The file's encoding must be UTF-8. The file should not start with a byte order marker (BOM), but if one exists, it will be skipped.

The rules for escaping special characters inside JSON are the ones in the official JSON specification, RFC 7159.

The command line '`mkvmerge -o "my file.mkv" -A "a movie.avi" sound.ogg`' could be converted into the following JSON option file called e.g. '`options.json`':

```
[
  "-o",
  "c:\\Matroska\\my file.mkv",
  "--title",
  "#65",
  "-A",
  "a movie.avi",
  "sound.ogg"
]
```

9. File linking

Matroska(tm) 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*(tm) segments. As most files will probably only contain one *Matroska*(tm) 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 timestamps 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 timestamps 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 SID in 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.

10. Default values

The *Matroska*(tm) 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.

11. Attachments

Maybe you also want to keep some photos along with your *Matroska*(tm) file, or you're using SSA subtitles and need a special *TrueType*(tm) font that's really rare. In these cases you can attach those files to the *Matroska*(tm) 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*(tm) fonts' case).

Here's an example how to attach a photo and a *TrueType*(tm) 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-mime-type application/octet-stream \
  --attach-file really_cool_font.ttf
```

If a *Matroska*(tm) 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.

12. Chapters

The *Matroska*(tm) 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 of *Matroska*(tm)'s chapter functionality.

Apart from dedicated chapter files *mkvmerge*(1) can also read chapters from other file formats (e.g. MP4, Ogg, Blu-rays or DVDs).

12.1. The simple chapter format

This format consists of pairs of lines that start with '`CHAPTERxx=`' and '`CHAPTERxxNAME=`' respectively. The first one contains the start timestamp 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*(tm) `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.

12.2. 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.

The following lists the supported XML tags, their data types and, where appropriate, the valid range for their values:

```
Chapters (master)
  EditionEntry (master)
    EditionUID (unsigned integer, valid range: 1 <= value)
    EditionFlagHidden (unsigned integer, valid range: 0 <= value <= 1)
    EditionFlagDefault (unsigned integer, valid range: 0 <= value <= 1)
    EditionFlagOrdered (unsigned integer, valid range: 0 <= value <= 1)
    ChapterAtom (master)
      ChapterAtom (master)
      ChapterUID (unsigned integer, valid range: 1 <= value)
      ChapterTimeStart (unsigned integer)
      ChapterTimeEnd (unsigned integer)
      ChapterFlagHidden (unsigned integer, valid range: 0 <= value <= 1)
      ChapterFlagEnabled (unsigned integer, valid range: 0 <= value <= 1)
      ChapterSegmentUID (binary, valid range: 1 <= length in bytes)
      ChapterSegmentEditionUID (unsigned integer, valid range: 1 <= value)
      ChapterPhysicalEquiv (unsigned integer)
      ChapterTrack (master)
        ChapterTrackNumber (unsigned integer, valid range: 1 <= value)
      ChapterDisplay (master)
        ChapterString (UTF-8 string)
        ChapterLanguage (UTF-8 string)
        ChapterCountry (UTF-8 string)
      ChapterProcess (master)
        ChapterProcessCodecID (unsigned integer)
        ChapterProcessPrivate (binary)
        ChapterProcessCommand (master)
          ChapterProcessTime (unsigned integer)
          ChapterProcessData (binary)
```

12.3. Reading chapters from Blu-rays

mkvmerge(1) can read chapters from unencrypted Blu-rays. For that you can use the path to one of the MPLS play lists with the *--chapters* parameter.

Example: `--chapters /srv/blurays/BigBuckBunny/BDMV/PLAYLIST/00001.mpls`

12.4. Reading chapters from DVDs

When MKVToolNix is compiled with the *libdvdread*(tm) library, *mkvmerge*(1) can read chapters from DVDs. For that you can use the path to one of the folders or files on the DVD with the *--chapters* parameter. As DVDs can contain more than one title and each title has its own set of chapters, you can append a colon and the desired title number to the end of the file name argument. The title number defaults to 1.

Example: `--chapters /srv/dvds/BigBuckBunny/VIDEO_TS:2`

12.5. 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 timestamps will be offset to match the new timestamps of each output file.

mkvmerge(1) is able to copy chapters from *Matroska*(tm) source files unless this is explicitly disabled with the --no-chapters option. The chapters from all sources (*Matroska*(tm) 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*(tm) 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.

13. Tags

13.1. Introduction

Matroska(tm)'s tag system is similar to that of other containers: a set of *KEY=VALUE* pairs. However, in *Matroska*(tm) 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 system.

13.2. Scope of the tags

Matroska(tm) 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 Matroska specification gives more details about this fact.

One important fact is that tags are linked to tracks or chapters with the `Targets` *Matroska*(tm) 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*(tm)) or which are copied from the source file if the track's source file is a *Matroska*(tm) 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*(tm) 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.

13.3. 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
```

13.4. 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.

13.5. Data types

The new *Matroska*(tm) 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*(tm) 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*(tm) tag specs. As *mkvmerge*(1) does not support this system anymore these types aren't described here.

13.6. Known tags for the XML file format

The following lists the supported XML tags, their data types and, where appropriate, the valid range for their values:

```
Tags (master)
  Tag (master)
    Targets (master)
      TargetTypeValue (unsigned integer)
      TargetType (UTF-8 string)
      TrackUID (unsigned integer)
      EditionUID (unsigned integer)
      ChapterUID (unsigned integer)
      AttachmentUID (unsigned integer)
    Simple (master)
      Simple (master)
      Name (UTF-8 string)
      TagLanguage (UTF-8 string)
      DefaultLanguage (unsigned integer)
      String (UTF-8 string)
      Binary (binary)
```

14. The segment info XML files

With a segment info XML file it is possible to set certain values in the "segment information" header field of a *Matroska*(tm) file. All of these values cannot be set via other command line options.

Other "segment information" header fields can be set via command line options but not via the XML file. This includes e.g. the `--title` and the `--timestamp-scale` options.

There are other elements that can be set neither via command line options nor via the XML files. These include the following elements: `DateUTC` (also known as the "muxing date"), `MuxingApp`, `WritingApp` and `Duration`. They're always set by *mkvmerge*(1) itself.

The following lists the supported XML tags, their data types and, where appropriate, the valid range for their values:

```
Info (master)
  SegmentUID (binary, valid range: length in bytes == 16)
  SegmentFilename (UTF-8 string)
  PreviousSegmentUID (binary, valid range: length in bytes == 16)
  PreviousSegmentFilename (UTF-8 string)
  NextSegmentUID (binary, valid range: length in bytes == 16)
  NextSegmentFilename (UTF-8 string)
  SegmentFamily (binary, valid range: length in bytes == 16)
  ChapterTranslate (master)
    ChapterTranslateEditionUID (unsigned integer)
    ChapterTranslateCodec (unsigned integer)
    ChapterTranslateID (binary)
```

15. Matroska file layout

The *Matroska*(tm) 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*(tm) file would look like this:

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

This might be the case for audio-only files.

16. External timestamp files

mkvmerge(1) allows the user to chose the timestamps 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*(tm) block. For video this is exactly one frame, for audio this is one packet of the specific audio type. E.g. for AC-3 this would be a packet containing `1536` samples.

Timestamp 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 timestamps then your command line

must look something like this:

```
$ mkvmerge ... --timestamps 0:my_timestamps.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.

16.1. Timestamp 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:

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

16.2. Timestamp file format v2

In this format each line contains a timestamp for the corresponding frame. This timestamp 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 timestamp lines as there are frames in the track. The timestamps in this file must be sorted. Example for 25fps:

```
# timestamp format v2
0
40
80
```

16.3. Timestamp 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 timestamps 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:

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

16.4. Timestamp file format v4

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

17. Exit codes

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

- `0` -- This exit code 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.

18. Environment variables

mkvmerge(1) uses the default variables that determine the system's locale (e.g. `LANG` and the `LC_*` family). Additional variables:

Option	Description
`MKVMERGE_DEBUG`, `MKVTOOLNIX_DEBUG` and its short form `MTX_DEBUG`	The content is treated as if it had been passed via the --debug option.
`MKVMERGE_ENGAGE`, `MKVTOOLNIX_ENGAGE` and its short form `MTX_ENGAGE`	The content is treated as if it had been passed via the --engage option.

19. See also

mkvinfo(1), *mkvextract*(1), *mkvpropedit*(1), *mkvtoolnix-gui*(1)

20. WWW

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