NAME

shntool − a multi-purpose WAVE data processing and reporting utility

SYNOPSIS

shntool *mode ...*
shntool [*CORE OPTION*]

DESCRIPTION

shntool is a command-line utility to view and/or modify WAVE data and properties. It runs in several different operating modes, and supports various lossless audio formats.

shntool is comprised of three parts - its core, *mode* modules, and *format* modules. This helps to make the code easier to maintain, as well as aid other programmers in developing new functionality. The distribution archive contains a file named 'modules.howto' that describes how to create a new mode or format module, for those so inclined.

Mode modules

shntool performs various functions on WAVE data through the use of mode modules. The core of **shntool** is simply a wrapper around the mode modules. In fact, when **shntool** is run with a valid mode as its first argument, it essentially runs the main procedure for the specified mode, and quits. **shntool** comes with several built-in modes, described below:

len	Displays length, size and properties of PCM WAVE data
fix	Fixes sector-boundary problems with CD-quality PCM WAVE data
hash	Computes the MD5 or SHA1 fingerprint of PCM WAVE data
pad	Pads CD(hyquality files not aligned on sector boundaries with silence
join	Joins PCM WAVE data from multiple files into one
split	Splits PCM WAVE data from one file into multiple files
cat	Writes PCM WAVE data from one or more files to the terminal
cmp	Compares PCM WAVE data in two files
cue	Generates a CUE sheet or split points from a set of files
conv	Converts files from one format to another
info	Displays detailed information about PCM WAVE data
strip	Strips extra RIFF chunks and/or writes canonical headers
gen	Generates CD-quality PCM WAVE data files containing silence
trim	Trims PCM WAVE silence from the ends of files

For more information on the meaning of the various command-line options for each mode, see the **MODE-SPECIFIC OPTIONS** section below.

For convenience, each mode can specify an alternate name or alias that will invoke it (this feature is currently only available on systems that support symbolic or hard linking). In particular, each mode is aliased to 'shn<mode>'. For instance, running *shnlen* is equivalent to running **shntool** *len* - thus saving a few keystrokes.

Format modules

File formats are abstracted from **shntool** through the use of format modules. They provide a means for **shntool** to tranparently read and/or write different file formats. This abstraction allows **shntool** to

concentrate on its job without worrying about the details of each file format.

The following formats are currently supported:

wav RIFF WAVE file format

aiff Audio Interchange File Format (AIFF and uncompressed/sowt AIFF−C only) (via 'sox'):
<http://sox.sourceforge.net/>

shn Shorten low complexity waveform coder (via 'shorten'):
<http://www.softsound.com/Shorten.html>
<http://www.etree.org/shnutils/shorten/>

flac Free Lossless Audio Codec (via 'flac'):
<http://flac.sourceforge.net/>

ape Monkey's Audio Compressor (via 'mac'):
<http://www.monkeysaudio.com/>
<http://supermmx.org/linux/mac/>

alac Apple Lossless Audio Codec (via 'alac'):
<http://craz.net/programs/itunes/alac.html>

tak (T)om's lossless (A)udio (K)ompressor (via 'takc'):
<http://www.thbeck.de/Tak/Tak.html>

ofr OptimFROG Lossless WAVE Audio Coder (via 'ofr'):
<http://www.losslessaudio.org/>

tta TTA Lossless Audio Codec (via 'ttaenc'):
<http://tta.sourceforge.net/>

als MPEG−4 Audio Lossless Coding (via 'mp4als'):
<http://www.nue.tu−berlin.de/forschung/projekte/ -
lossless/mp4als.html>

wv WavPack Hybrid Lossless Audio Compression (via 'wavpack' and 'wvunpack'):
<http://www.wavpack.com/>

lpac Lossless Predictive Audio Compression (via 'lpac'):
<http://www.nue.tu−berlin.de/wer/liebchen/lpac.html>

la Lossless Audio (via 'la'):
<http://www.lossless−audio.com/>

bonk Bonk lossy/lossless audio compressor (via 'bonk'):
<http://www.logarithmic.net/pfh/bonk>

kxs Kexis lossless WAV file compressor (via 'kexis'):
<http://www.sourceforge.net/projects/kexis/>

mkw MKW Audio Compression format (via 'mkwcon'):
<http://www.etree.org/shnutils/mkwcon/>

cust Custom output format module (output only, useful for encoding to a format that **shntool** does not yet support)

term sends output to the terminal

null sends output to /dev/null (output only, useful for dry-runs in several modes, such as *fix* mode or *strip* mode)

When reading files for input, **shntool** automatically discovers which, if any, format module handles each file. In modes where files are created as output, you can specify what the output format should be - otherwise, **shntool** decides for you by selecting the first format module it finds that supports output (in a default

installation, this will be the *wav* format).

CORE OPTIONS

Modeless

When run without a mode, **shntool** takes these options:

−m Show detailed mode module information

−f Show detailed format module information

−a Show default format module arguments

−v Show version information

−h Show a help screen

GLOBAL OPTIONS

All modes

All modes support the following options:

−D Print debugging information

−F *file* Specify a file containing a list of filenames to process. This overrides any files specified on the command line or on the terminal.

NOTE: Most modes will accept input filenames from a single source, according to the following order of precedence: file specified by the **−F** option, otherwise filenames on the command line, otherwise filenames read from the terminal.

−H Print times in h:mm:ss.{ff,nnn} format, instead of m:ss.{ff,nnn}

−P *type*

Specify progress indicator type. *type* is one of: {*pct*, *dot*, *spin*, *face*, *none*}. *pct* shows the completion percentage of each operation. *dot* shows the progress of each operation by displaying a '.' after each 10% step toward completion. *spin* shows a spinning progress indicator. *face* shows the progress of each operation by displaying six emoticons that become increasingly happy as the operation nears completion. *none* prevents any progress completion information from being displayed. The default is *pct*.

−h Show the help screen for this mode

−i *fmt* Specify input file format decoder and/or arguments. The format is: "fmt decoder [arg1 ... argN]", and must be surrounded by quotes. If arguments are given, then one of them must contain "%f", which will be replaced with the input filename. Examples:

−i '*shn shorten−2.3b*' (use official shorten−2.3b instead of later versions; leave default arguments untouched)

−i '*shn shorten −x −d 2048 %f −*' (force shorten to skip the first 2048 bytes of each file)

−q Suppress non-critical output (quiet mode). Output that normally goes to stderr will not be displayed, other than errors or debugging information (if specified).

−r *val* Reorder input files? *val* is one of: {*ask*, *ascii*, *natural*, *none*}. The default is *natural*.

−v Show version information

−w Suppress warnings

−− Indicates that everything following it is a filename

Output modes

Additionally, any mode that creates output files supports the the following options:

−O *val* Overwrite existing files? *val* is one of: {*ask*, *always*, *never*}. The default is *ask*.

−a *str* Prefix *str* to base part of output filenames

−d *dir* Specify output directory

−o *str* Specify output file format extension, encoder and/or arguments. Format is: "fmt [ext=abc] [encoder [arg1 ... argN (%f = filename)]]", and must be surrounded by quotes. If arguments are given, then one of them must contain "%f", which will be replaced with the output filename. Examples:

−o *'shn shorten −v2 − %f'* (create shorten files without seek tables)

−o *' flac flake − %f'* (use alternate flac encoder)

−o *'aiff ext=aif'* (override default aiff extension of 'aiff' with 'aif')

−o *'cust ext=mp3 lame −−quiet − %f'* (create mp3 files using lame)

−z *str* Postfix *str* to base part of output filenames

MODE-SPECIFIC OPTIONS

len mode options

−U *unit*

Specifies the unit in which the totals will be printed. *unit* is one of: {*b*, *kb*, *mb*, *gb*, *tb*}. The default is *b*.

−c Do not show column names

−t Do not show totals line

−u *unit* Specifies the unit in which each file will be printed. *unit* is one of: {*b*, *kb*, *mb*, *gb*, *tb*}. The default is *b*.

len mode output

The output of len mode may seem cryptic at first, because it attempts to convey a lot of information in just a little bit of space. But it is quite easy to read once you know what the columns represent; and in certain columns, what each character in the column means. Each column is explained below.

length Shows the length of the WAVE data, in m:ss.nnn (millisecond) format. If the data is CD-quality, then m:ss.ff is shown instead, where ff is a number from 00 to 74 that best approximates the number of frames (2352-byte blocks) remaining after m:ss. If all files are CD-quality, the total length will be shown in m:ss.ff format; otherwise it will be in m:ss.nnn format. NOTE: CD-quality files are rounded to the nearest frame; all other files are rounded to the nearest millisecond.

expanded size

Shows the total size of all WAVE chunks within the file (header, data and any extra RIFF chunks). Essentially this is the size that the file would be if it were converted to .wav format, e.g. with **shntool** *conv*.

NOTE: Do not rely on this field for audio size! If you simply want to know how many bytes of audio are in a file, run it through *info* mode, and look at the "data size" field in its output.

cdr Shows properties related to CD-quality files. A ’c’ in the first slot indicates that the WAVE data is not [C]D-quality. A ’b’ in the second slot indicates that the CD-quality WAVE data is not cut on a sector [b]oundary. An ’s’ in the third slot indicates that the CD-quality WAVE data is too [s]hort to be burned.

A ’−’ in any of these slots indicates that the particular property is OK or normal. An ’x’ in any of these slots indicates that the particular property does not apply to this file, or cannot be determined.

WAVE Shows properties of the WAVE data. An ’h’ in the first slot indicates that the WAVE [h]eader is not canonical. An ’e’ in the second slot indicates that the WAVE file contains [e]xtra RIFF chunks.

A ’−’ in any of these slots indicates that the particular property is OK or normal. An ’x’ in any of these slots indicates that the particular property does not apply to this file, or cannot be determined.

problems
Shows problems detected with the WAVE header, WAVE data, or the file itself. A ’3’ in the first slot indicates that the file contains an ID[3]v2 header. An ’a’ in the second slot indicates that the audio data is not block-[a]ligned. An ’i’ in the third slot indicates that the WAVE header is [i]nconsistent about data size and/or file size. A ’t’ in the fourth slot indicates that the WAVE file seems to be [t]runcated. A ’j’ in the fifth slot indicates that the WAVE file seems to have [j]unk appended to it.

A ’−’ in any of these slots indicates that the particular problem was not detected. An ’x’ in any of these slots indicates that the particular problem does not apply to this file, or cannot be determined.

fmt Shows which file format handled this file.

ratio Shows the compression ratio for this file.

filename
Shows the name of the file that’s being inspected.

fix mode options

NOTE: file names for files created in *fix* mode will be based on the input file name with the string ’−fixed’ appended to it, and the extension will be the default extension of the output file format. For example, with an output file format of *shn* the file ’foo.wav’ would become ’foo−fixed.shn’. This can be overridden with the **−a** and/or **−z** global options described above.

−b Shift track breaks backward to the previous sector boundary. This is the default.

−c Check whether fixing is needed, without actually fixing anything. **shntool** will exit with status 0 if fixing is needed, and status 1 otherwise. This can be useful in shell scripts, e.g.: "if shntool fix −c *; then shntool fix *; else ...; fi"

−f Shift track breaks forward to the next sector boundary.

−k Specifies that all files should be processed, even if the first several of them wouldn’t be altered, aside from a possible file format change. The default is to skip the first N files that wouldn’t be changed from a WAVE data perspective in order to avoid unnecessary work.

−n Specifies that the last file created should not be padded with silence to make its WAVE data size a multiple of 2352 bytes. The default is to pad the last file.

−u Round track breaks to the nearest sector boundary.

hash mode options

−c Specifies that the composite fingerprint for all input files should be generated, instead of the default of one fingerprint per file. The composite fingerprint is simply the fingerprint of the WAVE data from all input files taken as a whole in the order given, and is identical to the one that would be generated from the joined file if the same files were joined into one large file, with no padding added. This option can be used to fingerprint file sets, or to identify file sets in which track breaks have been moved around, but no audio has been modified in any way (e.g. no padding added, no resampling done, etc.).

−m Generate MD5 fingerprints. This is the default.

−s Generate SHA1 fingerprints.

pad mode options

NOTE: file names for files created in *pad* mode will be based on the input file name with the string '−prepadded' or '−postpadded' appended to it, and the extension will be the default extension of the output file format. For example, with an output file format of *shn* and pre-padding specified on the command line, the file 'foo.wav' would become 'foo−prepadded.shn'. This can be overridden with the **−a** and/or **−z** global options described above.

Be aware that some output format encoders (e.g. flac, ape) automatically strip headers and/or extra RIFF chunks.

−b Specifies that the file created should be padded at the beginning with silence to make its WAVE data size a multiple of 2352 bytes.

−e Specifies that the file created should be padded at the end with silence to make its WAVE data size a multiple of 2352 bytes. This is the default action.

join mode options

NOTE: file names for files created in *join* mode will be prefixed with 'joined.', and the extension will be the default extension of the output file format. For example, with an output file format of *wav* the files 'files*.wav' would become 'joined.wav'. This can be overridden with the **−a** and/or **−z** global options described above.

−b Specifies that the file created should be padded at the beginning with silence to make its WAVE data size a multiple of 2352 bytes. Note that this option does not apply if the input files are not CD-quality, since padding is undefined in that case.

−e Specifies that the file created should be padded at the end with silence to make its WAVE data size a multiple of 2352 bytes. This is the default action. Note that this option does not apply if the input files are not CD-quality, since padding is undefined in that case.

−n Specifies that the file created should not be padded with silence to make its WAVE data size a multiple of 2352 bytes. Note that this option does not apply if the input files are not CD-quality, since padding is undefined in that case.

split mode options

NOTE: file names for files created in *split* mode are of the form prefixNNN.ext, where NNN is the output file number, and 'ext' is the default extension of the output file format. If an output file format of 'wav' is used, and the prefix is not altered via the **−n** switch described below, then the output file names will be "split−track01.wav", "split−track02.wav", etc. This can be overridden with the **−a** and/or **−z** global options described above.

For information on specifying split points, see the **Specifying split points** section below.

−c *num* Specifies the number to start counting from when naming output files. The default is 1.

−e *len* Prefix each track with *len* amount of lead-in taken from the previous track. *len* must be given in bytes, m:ss, m:ss.ff or m:ss.nnn format.

−f *file* Specifies a file from which to read split point data. If not given, then split points are read from the terminal.

−l *len* Specifies that the input file should be split into smaller files based on multiples of the *len* time interval. *len* must be given in bytes, m:ss, m:ss.ff or m:ss.nnn format.

−m *str* Specifies a character manipulation string for filenames generated from CUE sheets. These characters, taken one-by-one, represent from/to character translation. They must always be in pairs. Some examples:

:− Translate all instances of ':' to '−'

:−/− Translate both ':' and '/' to '−'

*:−/_*x* Translate ':' to '−', '/' to '_', and '*' to 'x'

−n *fmt* Specifies the file count output format. The default is %02d, which gives two-digit zero-padded numbers (01, 02, 03, ...).

−t *fmt* Name output files in user-specified format based on CUE sheet fields. The following formatting strings are recognized:

%p Performer

%a Album

%t Track title

%n Track number

−u *len* Postfix each track with *len* amount of lead-out taken from the next track. *len* must be given in bytes, m:ss, m:ss.ff or m:ss.nnn format.

−x *list* Only extract tracks in *list* (comma separated, may contain ranges). Examples include:

7 Only extract track 7

3−5 Only extract tracks 3 through 5

2−6,9,11−13
Only extract tracks 2 through 6, 9, and 11 through 13

Specifying split points

Split points simply mark places within the WAVE data of the input file where tracks will be split. They can be specified in any combination of the following formats:

bytes where bytes is a specific byte offset

m:ss where m = minutes and ss = seconds

m:ss.ff where m = minutes, ss = seconds and ff = frames (75 per second, so ff ranges from 00 to 74)

m:ss.nnn
where m = minutes, ss = seconds and nnn = milliseconds (will be rounded to closest sector boundary, or the first sector boundary if the closest one happens to be the beginning of the file)

CUE sheet
- a simple CUE sheet, in which each "INDEX 01 m:ss:ff" line is converted to a m:ss.ff split point

Split points must be given in increasing order, and must appear one per line. If the byte offset calculated from the final split point equals the input file's WAVE data size, then it is ignored. Since split points specify locations within the input file where tracks will be split, N split points will create N+1 output files. All m:ss formats will create splits on sector boundaries whenever the input file is CD-quality; to force non-sector-aligned splits, use the exact byte format.

cat mode options

−c Specifies that extra RIFF chunks should be suppressed from the output. The default is to write the extra RIFF chunks.

−d Specifies that the WAVE data should be suppressed from the output. The default is to write the data.

−e Specifies that the WAVE header should be suppressed from the output. The default is to write the header.

−n Specifies that the NULL pad byte at end of odd−sized data chunks should be suppressed from the output, if present. The default is to write the NULL pad byte. This option only applies when WAVE data is also written, otherwise it is ignored.

cmp mode options

−c *secs* Sets the number of seconds of audio to use for the byte-shift comparison buffer. This option only makes sense with the **−s** option. The default is 3 seconds.

−f *fuzz* Sets the "fuzz factor" for determining whether byte-shifted data is identical. *fuzz* is a positive integer that represents the maximum number of allowable byte mismatches between the two files in the area searched by the **−s** option. This allows one to check for differing bytes between to files that (a) are byte-shifted and (b) contain at least one error in the area searched by the **−s** option. The higher the fuzz factor, the longer the search takes, so set it low to begin with (8 or so), and increase it in small steps if needed. NOTE: this switch can only be used with the **−s** switch.

−l List offsets and values of all differing bytes. Output is similar to 'cmp −l'; in particular, offsets are 1-based. Can be used with the **−s** switch.

−s Check to see whether the WAVE data contained in the input files are identical modulo a byte-shift. Currently, this will only detect differences up to the first 529200 bytes (equal to 3 seconds of CD-quality data). This can be used to compare WAVE data within a pre-burned file to WAVE data in the corresponding track ripped from the burned CD, which is useful if the ripped track came from a CD burned TAO, and thus might have a 2-second gap of silence at the beginning. This option can also help identify a CD burner/CD reader combined read/write offset.

cue mode options

−c Specifies that a simple CUE sheet should be output. This is the default action. NOTE: all input files must be CD-quality for CUE sheets to be valid.

−s Specifies that split points in explicit byte-offset format should be output.

conv mode options

NOTE: file names for files created in *conv* mode will be named based on the input file name. Specifically, if the input file name ends with the default file extension for that file's format, then the default extension for the desired output format will replace it; otherwise, it will be appended to it. For example, for an output format of *shn* and a *wav* input file named 'file.wav', the converted file will be named 'file.shn', since '.wav' is the default extension for the *wav* format. On the other hand, given the same situation above, but with an input file named 'file.wave', the converted file will be named 'file.wave.shn', since '.wave' does not match '.wav'. This can be overridden with the **−a** and/or **−z** global options described above.

Be aware that some output format encoders (e.g. flac, ape) automatically strip headers and/or extra RIFF chunks, while others (e.g. sox) might adjust WAVE data sizes in rare instances in order to align the audio on

a block boundary.

−t Read WAVE data from the terminal.

info mode options

This mode doesn't support any additional options.

strip mode options

NOTE: file names for files created in *strip* mode will be based on the input file name with the string '−stripped' appended to it, and the extension will be the default extension of the output file format. For example, with an output file format of *wav* the file 'bar.shn' would become 'bar−stripped.wav'. This can be overridden with the **−a** and/or **−z** global options described above.

Be aware that some output format encoders (e.g. flac, ape) automatically strip headers and/or extra RIFF chunks, while others (e.g. sox) might adjust WAVE data sizes in rare instances in order to align the audio on a block boundary.

−c Specifies that extra RIFF chunks should not be stripped. The default is to remove everything that appears after the first data chunk.

−e Specifies that WAVE headers should not be made canonical. The default is to canonicalize headers.

gen mode options

NOTE: file names for files created in *gen* mode will be prefixed with 'silence.', and the extension will be the default extension of the output file format. For example, with an output file format of *wav* the generated file would become 'silence.wav'. This can be overridden with the **−a** and/or **−z** global options described above.

−l *len* Generate files containing *len* amount of silence. *len* must be given in bytes, m:ss, m:ss.ff or m:ss.nnn format.

trim mode options

NOTE: file names for files created in *trim* mode will be based on the input file name with the string '−trimmed' appended to it, and the extension will be the default extension of the output file format. For example, with an output file format of *shn* the file 'foo.wav' would become 'foo−trimmed.shn'. This can be overridden with the **−a** and/or **−z** global options described above.

−b Only trim silence from the beginning of files

−e Only trim silence from the end of files

ENVIRONMENT VARIABLES

ST_DEBUG

If set, shntool will print debugging information. This is analogous to the **−D** global option, with the exception that debugging is enabled immediately, instead of when the command-line is parsed.

ST_<FORMAT>_DEC

Specify input file format decoder and/or arguments. Replace **<FORMAT>** with the format you wish to modify, e.g. **ST_SHN_DEC**. The format of this variable is analagous to the **−i** global option, except that the initial format is not included. Examples:

ST_SHN_DEC='**shorten−2.3b**'

ST_SHN_DEC='**shorten −x −d 2048 %f −**'

ST_<FORMAT>_ENC

Specify output file format extension, encoder and/or arguments. Replace **<FORMAT>** with the format you wish to modify, e.g. **ST_SHN_ENC**. The format of this variable is analagous to the **−o** global option, except that the initial format is not included. Examples:

ST_SHN_ENC=’**shorten −v2 − %f**’

ST_FLAC_ENC=’**flake − %f**’

ST_AIFF_ENC=’**ext=aif**’

ST_CUST_ENC=’**ext=mp3 lame −−quiet − %f**’

Note that command-line options take precedence over any of these environment variables.

EXIT STATUS

Generally speaking, **shntool** will exit with status 0 upon success, and status 1 if it encounters an error. The only exception is when the ’quit’ option is selected from within the interactive file reordering menu, in which case the exist status will be 255.

NOTES

shntool is a misnomer, since it processes WAVE data, not shorten data. The name is a holdover from its early days as ’shnlen’, a program created specifically to extract information about WAVE data stored within .shn files.

Aliases for **shntool** are prefixed with ’shn’ instead of ’wav’ to avoid possible collisions with existing programs.

AUTHOR

Jason Jordan <shnutils at freeshell dot org>

Please send all bug reports to the above address.

The latest version of **shntool** can always be found at <http://www.etree.org/shnutils/> or <http://shnutils.freeshell.org/>.

COPYRIGHT

REVISION

$Id: shntool.1,v 1.140 2009/03/30 05:59:25 jason Exp $