NAME

shorten − fast compression for waveform files

SYNOPSIS

shorten [-hlu] [-a #bytes] [-b #samples] [-c #channels] [-d #bytes] [-m #blocks] [-n #dB] [-p #order] [-q #bits] [-r #bits] [-t filetype] [-v #version] [waveform-file [shortened-file]]

shorten -x [-hl] [-a #bytes] [-d #bytes] [shortened-file [waveform-file]]

shorten [-e | -i | -k | -s | -S<name>] shortened-file

shorten [-s | -S<name>] < shortened-data

DESCRIPTION

shorten reduces the size of waveform files (such as audio) using Huffman coding of prediction residuals and optional additional quantisation. In lossless mode the amount of compression obtained depends on the nature of the waveform. Those composing of low frequencies and low amplitudes give the best compression, which may be 2:1 or better. Lossy compression operates by specifying a minimum acceptable segmental signal to noise ratio or a maximum bit rate. Lossy compression operates by zeroing the lower order bits of the waveform, so retaining waveform shape.

If both file names are specified then these are used as the input and output files. The first file name can be replaced by "-" to read from standard input and likewise the second filename can be replaced by "-" to write to standard output. Under UNIX, if only one file name is specified, then that name is used for input and the output file name is generated by adding the suffix ".shn" on compression and removing the ".shn" suffix on decompression. In these cases the input file is removed on completion. The use of automatic file name generation is not currently supported under DOS. If no file names are specified, shorten reads from standard input and writes to standard output. Whenever possible, the output file inherits the permissions, owner, group, access and modification times of the input file.

From release 2.3 the RIFF WAVE (Microsoft .wav) file type is the default. These files contain enough information to set most of the switches presented below, so effective operation is obtained just by setting the desired level of compression (-n or -r switch).

OPTIONS

−a *align bytes*
: Specify the number of bytes to be copied verbatim before compression begins. This option can be used to preserve fixed length ASCII headers on waveform files, and may be necessary if the header length is an odd number of bytes.

−b *block size*
: Specify the number of samples to be grouped into a block for processing. Within a block the signal elements are expected to have the same spectral characteristics. The default option works well for a large range of audio files.

−c *channels*
: Specify the number of independent interwoven channels. For two signals, a(t) and b(t) the original data format is assumed to be a(0),b(0),a(1),b(1)...

−d *discard bytes*
: Specify the number of bytes to be discarded before compression or decompression. This may be used to delete header information from a file. Refer to the -a option for storing the header information in the compressed file.

−e Erase seek information from an existing file.

−h Give a short message specifying usage options.

−i Inquire as to whether the given file is an external seek table file, a file with seek tables appended to it, or neither. If seek tables are present, the seek table revision number is shown.

−k Append seek information to an existing file.

−l Prints the software license specifying the conditions for the distribution and usage of this software.

−m *blocks*
Specify the number of past blocks to be used to estimate the mean and power of the signal. The value of zero disables this prediction and the mean is assumed to lie in the middle of the range of the relevant data type (i.e. at zero for signed quantities). The default value is non-zero for format versions 2.0 and above.

−n *noise level*
Specify the minimum acceptable segmental signal to noise ratio in dB. The signal power is taken as the variance of the samples in the current block. The noise power is the quantisation noise incurred by coding the current block assuming that samples are uniformally distributed over the quantisation interval. The bit rate is dynamically changed to maintain the desired signal to noise ratio. The default value represents lossless coding.

−p *prediction order*
Specify the maximum order of the linear predictive filter. The default value of zero disables the use of linear prediction and a polynomial interpolation method is used instead. The use of the linear predictive filter generally results in a small improvement in compression ratio at the expense of execution time. This is the only option to use a significant amount of floating point processing during compression. Decompression still uses a minimal number of floating point operations.

Decompression time is normally about twice that of the default polynomial interpolation. For version 0 and 1, compression time is linear in the specified maximum order as all lower values are searched for the greatest expected compression (the number of bits required to transmit the prediction residual is monotonically decreasing with prediction order, but transmitting each filter coefficient requires about 7 bits). For version 2 and above, the search is started at zero order and terminated when the last two prediction orders give a larger expected bit rate than the minimum found to date. This is a reasonable strategy for many real world signals - you may revert back to the exhaustive algorithm by setting -v1 to check that this works for your signal type.

−q *quantisation level*
Specify the number of low order bits in each sample which can be discarded (set to zero). This is useful if these bits carry no information, for example when the signal is corrupted by noise.

−r *bit rate*
Specify the expected maximum number of bits per sample. The upper bound on the bit rate is achieved by setting the low order bits of the sample to zero, hence maximising the segmental signal to noise ratio.

−s Write seek table information to a separate file (uses shortened file name with ’.skt’ extension). If the shortened data is read from standard input, then the seek table information will be saved in ’stdin.skt’.

−S<*name*>
Write seek table information to a separate file given by "<name>".

−t *file type*
Gives the type of the sound sample file as one of aiff, wav, s8, u8, s16, u16, s16x, u16x, s16hl, u16hl, s16lh, u16lh, ulaw, or alaw.

The simple types are listed first and have an initial s or u for signed or unsigned data, followed by 8 or 16 as the number of bits per sample. No further extension means the data is in the natural byte order, a trailing x specifies byte swapped data, hl explicitly states the byte order as high byte followed by low byte and lh the converse. Hence s16 means signed 16 bit integers in the natural byte order (like C would fwrite() shorts).

ulaw is the natural file type of ulaw encoded files (such as the default sun .au files) and alaw is a similar byte-packed scheme. Specific optimisations are applied to ulaw and alaw files. If lossless compression is specified with ulaw files then a check is made that the whole dynamic range is used (useful for files recorded on a SparcStation with the volume set too high). Lossless coding of both file types uses an internal format with a monotonic mapping to linear. If lossy compression is specified then the data is internally converted to linear. The lossy option "-r4" has been observed to give little degradation and provides 2:1 compression.

With the types listed above you should explicitly set the number of channels (if not mono) with -c and if the file contains a header the size should be specified with -a. This is most important for lossy compression which will lead to data corruption if a file header is inadvertently lossy coded.

Finally, as of version 2.3, the file type may be specified as wav (the default). In this case the file to be compressed is interogated for the specific data type (chosen from the above) and the number of channels to be used. The header length alignment (-a flag) is also automatic so lossless compression requires no switches to be set and lossy compression requires only that the compression level be set with -n or -r.

−u The ulaw standard (ITU G711) has two codes which both map onto the zero value on a linear scale. The "-u" flag maps the negative zero onto the positive zero and so yields marginally better compression for format version 2 (the gain is significant for older format versions).

−v *version*
Specify the binary format version number of compressed files. Legal values are currently 1, 2 and 3, with higher numbers generally giving better compression. 2 and 3 are identical, with the exception that 2 does not generate seek tables, while 3 does. Detection of format version on decode is automatic.

−x *extract*
Reconstruct the original file. All other command line options except -a and -d are ignored.

METHODOLOGY

shorten works by blocking the signal, making a model of each block in order to remove temporal redundancy, then Huffman coding the quantised prediction residual.

Blocking

The signal is read in a block of about 128 or 256 samples, and converted to integers with expected mean of zero. Sample-wise-interleaved data is converted to separate channels, which are assumed independent.

Decorrelation

Four functions are computed, corresponding to the signal, difference signal, second and third order differences. The one with the lowest variance is coded. The variance is measured by summing absolute values for speed and to avoid overflow.

Compression

It is assumed the signal has the Laplacian probability density function of exp(-abs(x)). There is a computationally efficient way of mapping this density to Huffman codes, The code is in four parts: a run of zeros; a bounding one; a fixed number of bits mantissa; and the sign bit. The number of leading zeros gives the offset from zero. Some examples for a 2 bit mantissa:

Value	**zeros**	**stopbit**	**mantissa**	**signbit**	**total code**
0		**1**	**00**	**0**	**1000**
1		**1**	**01**	**0**	**1010**
2		**1**	**10**	**0**	**1010**
4	**0**	**1**	**00**	**0**	**01000**
7	**0**	**1**	**11**	**0**	**01110**
8	**00**	**1**	**00**	**0**	**001000**

-1		**1**	**00**	**1**	**1001**
-2		**1**	**01**	**1**	**1011**
-7	**0**	**1**	**10**	**1**	**01101**

Note that negative numbers are offset by one as there is no need to have two zero codes. The technical report CUED/F-INFENG/TR.156 included with the shorten distribution as files tr154.tex and tr154.ps contains bugs in this format description and is superceeded by this man page.

EMBEDDED OPERATION

Shorten may be used embedded within other programs. shorten is a function call implemented in the file shorten.c. The file main.c provides a wrapper for stand alone operation. A simple example of ebedded operation can be found in the file embedded.c. Full windows DLL operation is provided in the windll subdirectory.

SEE ALSO

compress(1),**pack**(1).

DIAGNOSTICS

Exit status is normally 0. A warning is issued if the file is not properly aligned, i.e. a whole number of records could not be read at the end of the file.

BUGS

An easy way to test shorten for your system is to use "make check", if this fails, for whatever reason, please report it to <shnutils@freeshell.org>.

No check is made for increasing file size, but valid waveform files generally achieve some compression. Even compressing a file of random bytes (which represents the worst case waveform file) only results in a small increase in the file length (about 6% for 8 bit data and 3% for 16 bit data). There is one condition that is know to be problematic, that is the lossy compression of unsigned data without mean estimation - large file sizes may result if the mean is far from the middle range value. For these files the value of the -m switch should be non-zero, as it is by default in format version 2.

There is no provision for different channels containing different data types. Normally, this is not a restriction, but it does mean that if lossy coding is selected for the ulaw type, then all channels use lossy coding.

The technical report CUED/F-INFENG/TR.156 (included in the shorten distribution) report contains errors in the bitfield format description and is superceeded by this document.

See the file "ChangeLog" for a history of bug fixes and feature additions.

Please mail Jason Jordan at the address below if you find a bug in shorten involving seek tables.

Please mail Brian Willoughby at the address below if you find a bug in the AIFF implementation.

Please mail Tony Robinson immediately at the address below if you find a bug in shorten that is NOT related to seek tables or AIFF support. Make sure you can reproduce your bug using version 2.3a, the last version known to be released by him.

AVAILABILITY

The latest 2.x and 3.x versions can be obtained from <http://www.etree.org/shnutils/shorten/> or <http://shnutils.freeshell.org/shorten/>.

AUTHORS

Copyright (C) 1992-1999 by Tony Robinson and SoftSound Ltd (ajr@softsound.com)

Unix maintenance of 3.x versions by Jason Jordan <shnutils@freeshell.org>.

AIFF support and maintenance by Brian Willoughby <shorten@sounds.wa.com> of Sound Consulting <http://sounds.wa.com/>.

Shorten is available for non-commercial use without fee. See the LICENSE file for the formal copying and usage restrictions. For supported versions please see http://www.softsound.com/Shorten.html and for commercial use please contact shorten@softsound.com