NAME

shuff − semi-static canonical coder for files of unsigned integers

SYNOPSIS

shuff −e [**−b** *block_size*] [**−Z**] [**−v** { 1 | 2 | 3 }] [*file*]

shuff −e1 *filename* [**−v** *n*] [*file*]

shuff −e2 *filename* [**−v** *n*] [*file*]

shuff −d [**−v** *n*] [*file*]

DESCRIPTION

shuff encodes and decodes a file of unsigned integers using semi-static canonical minimum-redundancy (Huffman) coding to *stdout*. There are two options for encoding: two-pass coding, which requires a pass over the input file to gather the frequencies of the integers, and a second pass to perform the actual coding; or one-pass coding where the integer file is broken into blocks and two passes are made on each block in memory (so the file is only read once).

Decoding is the same for files encoded with either method.

Two-pass encoding

The two-pass encoding process writes an auxiliary file of integer frequencies which is used as the *filename* argument of the -e1 and -e2 options. The file of frequencies is not required for decoding, since **shuff** stores a representation of the Huffman code within the compressed output.

−e1 *filename*
: Causes **shuff** to make an intial pass though *file* or *stdin*, writing a file of symbol frequencies to *filename*.

−e2 *filename*
: Causes **shuff** to read the symbol frequency information from *filename,* build a minimum-redundancy canonical prefix code for that distribution, and then pass through *file* or *stdin* reading unsigned integers, calculating their corresponding codewords, and writing encoded bits to *stdout*. The file *filename* should previously have been created from the same, or representative, data as is being compressed during the second pass.

−v *n*
: Prints statistical information to *stderr* showing the operations being performed. The larger the value of *n* the more detailed (and voluminous) the output.

One-pass encoding

One-pass encoding splits the *file* or *stdin* into blocks, and then applies the algorithms of the two-pass coder to each block in memory, writing bits to *stdout*. The blocks can be of fixed length (using the **-b** *blocksize* option), or can be terminated by integer zeroes in the input file (using the **−Z** option).

−e
: Use one pass encoding.

−b *block_size*
: Encodes symbols in blocks of *block_size*.

−Z
: Treats all symbols between zero symbols as a single block (the -b option is ignored). The zeroes are also included in the compressed message and reproduced by the decoder, allowing the process that writes the file of integers to determine where block boundaries should lie − useful when each integer has a different probability in each block. It is not strictly necessary to have a zero as the final symbol of the file, but a warning message will be printed if this is not the case.

−v *n*
: Outputs summary information. Use of *n=2* outputs information per block in bits per symbol, *n=3* outputs information per block in bits.

USAGE

To encode a file named *numbers* using two pass coding into a file *numbers-enc* and then decode to a file *numbers-dec* you would proceed as

shuff −e1 *freqs numbers*

shuff −e2 *freqs numbers* > *numbers-enc*

rm *freqs*

shuff −d *numbers-enc* > *numbers-dec*

The files *numbers* and *numbers-dec* should be the same. (Check with **cmp** *numbers numbers-dec*)

To encode *numbers* in a single pass using a block size of 1 MB (assuming 4-byte integers):

shuff −e −b *262144 numbers* > *numbers-enc*

shuff −d *numbers-enc* > *numbers-dec*

Again, the files *numbers* and *numbers-dec* should be the same.

To encode *numbers* in a single pass using zero symbols as block terminators:

shuff −e −Z *numbers* > *numbers-enc*

shuff −d *numbers-enc* > *numbers-dec*

cmp *numbers numbers-dec*

ORIGINS

shuff is based upon original work of the two authors, described in "On the Implementation of Minimum-Redundancy Prefix Codes", *IEEE Transactions on Communications*, 45(10):1200-1207, October 1997, and "Housekeeping for Prefix Coding", *IEEE Transactions on Communications*, 48(4):622-628, April 2000.

For more details of the implementation, see the two papers listed above, or the book *Compression and Coding Algorithms* A. Moffat and A. Turpin, Kluwer Academic Press, February 2002. Further information about this book is available at http://www.cs.mu.oz.au/caca/

We ask that, if you use this software to derive experimental results that are reported in any way, you cite the original work in which the underlying processes are described (by referencing either both of the two listed papers, or the book); and also acknowledge our authorship of the implementation you have used.

BUGS

shuff has not been extensively tested, and should be used for research purposes only. Portability is not guaranteed. There is no warranty, either express or implied, that it is fit for any purpose whatsoever, and neither the authors nor The University of Melbourne accept any responsibility for any consequences that may arise from your use of this software.

LICENCE

AUTHORS

Andrew Turpin* and Alistair Moffat, Department of Computer Science and Software Engineering, The University of Melbourne, Victoria 3010, Australia. Email: aht@cs.mu.oz.au, alistair@cs.mu.oz.au. (*Now at Curtin University, Perth, Australia, Email: andrew@computing.edu.au)