Files output by consist of a header line, followed by a number of body lines, and a trailer line. The command will ignore any lines preceding the header or following the trailer. Lines preceding a header must not, of course, look like a header. The header line starts with the word a space, a file mode (in octal), a space, and finally a string which names the file being encoded. The central engine of is a six-bit encoding function which outputs an character. The six bits to be encoded are treated as a small integer and added with the value for the space character (octal 40). The result is a printable character. In the case where all six bits to be encoded are zero, the character ` (octal 140) is emitted instead of what would normally be a space. The body of an encoded file consists of one or more lines, each of which may be a maximum of 86 characters long (including the trailing newline). Each line represents an encoded chunk of data from the input file and begins with a byte count, followed by encoded bytes, followed by a newline. The byte count is a six-bit integer encoded with the above function, representing the number of bytes encoded in the rest of the line. The method used to encode the data expands its size by 133% (described below). Therefore it is important to note that the byte count describes the size of the chunk of data before it is encoded, not afterwards. The six bit size of this number effectively limits the number of bytes that can be encoded in each line to a maximum of 63. While will not encode more than 45 bytes per line, will tolerate the maximum line size. The remaining characters in the line represent the data of the input file encoded as follows. Input data are broken into groups of three eight-bit bytes, which are then interpreted together as a 24-bit block. The first bit of the block is the lowest order bit of the first character, and the last is the highest order bit of the third character. This block is then broken into four six-bit integers which are encoded one by one starting from the first bit of the block. The result is a four character string for every three bytes of input data. Encoded lines of data continue in this manner until the input file is exhausted. The end of the body is signaled by an encoded line with a byte count of zero (the character `). Obviously, not every input file will be a multiple of three bytes in size. In these cases, will pad the remaining one or two bytes of data with garbage bytes until a three byte group is created. The byte count in a line containing garbage padding will reflect the actual number of bytes encoded, making it possible to convey how many bytes are garbage. The trailer line consists of on a line by itself. The interpretation of the format relies on properties of the character set and may not work correctly on non-ASCII systems. The file format appeared in