Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'cksum.1p' command

$ man cksum.1p

CKSUM(1P) POSIX Programmer's Manual CKSUM(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

cksum ? write file checksums and sizes

SYNOPSIS

cksum [file...]

DESCRIPTION

The cksum utility shall calculate and write to standard output a cyclic redundancy check (CRC) for each input file, and also write to standard output the number of octets in each file. The CRC used is based on the polynomial used for CRC error checking in the ISO/IEC 8802?3:1996 stan? dard (Ethernet).

The encoding for the CRC checksum is defined by the generating polyno? mial:

G(x)=x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1

Mathematically, the CRC value corresponding to a given file shall be defined by the following procedure:

1. The n bits to be evaluated are considered to be the coefficients of a mod 2 polynomial M(x) of degree n-1. These n bits are the bits

from the file, with the most significant bit being the most signif? icant bit of the first octet of the file and the last bit being the least significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral number of octets, followed by one or more octets representing the length of the file as a binary value, least significant octet first. The smallest number of octets capable of representing this integer shall be used.

2. M(x) is multiplied by x32 (that is, shifted left 32 bits) and di? vided by G(x) using mod 2 division, producing a remainder R(x) of degree ? 31.

3. The coefficients of R(x) are considered to be a 32-bit sequence.

4. The bit sequence is complemented and the result is the CRC.

OPTIONS

None.

OPERANDS

The following operand shall be supported:

file A pathname of a file to be checked. If no file operands are specified, the standard input shall be used.

STDIN

The standard input shall be used if no file operands are specified, and shall be used if a file operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise, the standard input shall not be used. See the INPUT FILES section.

INPUT FILES

The input files can be any file type.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of ck? sum:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er? ror.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

For each file processed successfully, the cksum utility shall write in the following format:

"%u %d %s\n", <checksum>, <# of octets>, <pathname>

If no file operand was specified, the pathname and its leading <space> shall be omitted.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values shall be returned:

0 All files were processed successfully.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

The cksum utility is typically used to quickly compare a suspect file against a trusted version of the same, such as to ensure that files transmitted over noisy media arrive intact. However, this comparison cannot be considered cryptographically secure. The chances of a damaged file producing the same CRC as the original are small; deliberate de? ception is difficult, but probably not impossible.

Although input files to cksum can be any type, the results need not be what would be expected on character special device files or on file types not described by the System Interfaces volume of POSIX.1?2017. Since this volume of POSIX.1?2017 does not specify the block size used when doing input, checksums of character special files need not process all of the data in those files.

The algorithm is expressed in terms of a bitstream divided into octets. If a file is transmitted between two systems and undergoes any data transformation (such as changing little-endian byte ordering to big-en? dian), identical CRC values cannot be expected. Implementations per? forming such transformations may extend cksum to handle such situa? tions.

EXAMPLES

None.

RATIONALE

The following C-language program can be used as a model to describe the algorithm. It assumes that a char is one octet. It also assumes that the entire file is available for one pass through the function. This was done for simplicity in demonstrating the algorithm, rather than as an implementation model.

```
static unsigned long crctab[] = {
0x00000000,
0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b,
0x1a864db2, 0x1e475005, 0x2608edb8, 0x22c9f00f, 0x2f8ad6d6,
0x2b4bcb61, 0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbdbd,
0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9, 0x5f15adac,
0x5bd4b01b, 0x569796c2, 0x52568b75, 0x6a1936c8, 0x6ed82b7f,
```

0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3, 0x709f7b7a,
0x745e66cd, 0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,
0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5, 0xbe2b5b58,
0xbaea46ef, 0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033,
0xa4ad16ea, 0xa06c0b5d, 0xd4326d90, 0xd0f37027, 0xddb056fe,
0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1, 0xe13ef6f4,
0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d, 0x34867077, 0x30476dc0,
0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c, 0x2e003dc5,
0x2ac12072, 0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcdbb16,
0x018aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca, 0x7897ab07,
0x7c56b6b0, 0x71159069, 0x75d48dde, 0x6b93dddb, 0x6f52c06c,
0x6211e6b5, 0x66d0fb02, 0x5e9f46bf, 0x5a5e5b08, 0x571d7dd1,
0x53dc6066, 0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,
0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e6e04e, 0xbfa1b04b,
0xbb60adfc, 0xb6238b25, 0xb2e29692, 0x8aad2b2f, 0x8e6c3698,
0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044, 0x902b669d,
0x94ea7b2a, 0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e,
0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2, 0xc6bcf05f,
0xc27dede8, 0xcf3ecb31, 0xcbffd686, 0xd5b88683, 0xd1799b34,
0xdc3abded, 0xd8fba05a, 0x690ce0ee, 0x6dcdfd59, 0x608edb80,
0x644fc637, 0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,
0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f, 0x5c007b8a,
0x58c1663d, 0x558240e4, 0x51435d53, 0x251d3b9e, 0x21dc2629,
0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5, 0x3f9b762c,
0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,
0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623, 0xf12f560e,
0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65,
0xeba91bbc, 0xef68060b, 0xd727bbb6, 0xd3e6a601, 0xdea580d8,
0xda649d6f, 0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,
0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7, 0xae3afba2,
0xaafbe615, 0xa7b8c0cc, 0xa379dd7b, 0x9b3660c6, 0x9ff77d71,
0x92b45ba8, 0x9675461f, 0x8832161a, 0x8cf30bad, 0x81b02d74,

```
0x857130c3, 0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640,
0x4e8ee645, 0x4a4ffbf2, 0x470cdd2b, 0x43cdc09c, 0x7b827d21,
0x7f436096, 0x7200464f, 0x76c15bf8, 0x68860bfd, 0x6c47164a,
0x61043093, 0x65c52d24, 0x119b4be9, 0x155a565e, 0x18197087,
0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,
0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088, 0x2497d08d,
0x2056cd3a, 0x2d15ebe3, 0x29d4f654, 0xc5a92679, 0xc1683bce,
0xcc2b1d17, 0xc8ea00a0, 0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb,
0xdbee767c, 0xe3a1cbc1, 0xe760d676, 0xea23f0af, 0xeee2ed18,
0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4, 0x89b8fd09,
0x8d79e0be, 0x803ac667, 0x84fbdbd0, 0x9abc8bd5, 0x9e7d9662,
0x933eb0bb, 0x97ffad0c, 0xafb010b1, 0xab710d06, 0xa6322bdf,
0xa2f33668, 0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4
};
unsigned long memcrc(const unsigned char *b, size_t n)
{
/*  Input arguments:
 *  const unsigned char*   b == byte sequence to checksum
 *  size_t                 n == length of sequence
 */
    register size_t i;
    register unsigned c, s = 0;
    for (i = n; i > 0; --i) {
        c = *b++;
        s = (s << 8) ^ crctab[(s >> 24) ^ c];
    }
    /* Extend with the length of the string. */
    while (n != 0) {
        c = n & 0377;
        n >>= 8;
        s = (s << 8) ^ crctab[(s >> 24) ^ c];
    }
    return ~s;
```

```
}
```

The historical practice of writing the number of ``blocks'' has been changed to writing the number of octets, since the latter is not only more useful, but also since historical implementations have not been consistent in defining what a ``block'' meant.

The algorithm used was selected to increase the operational robustness of cksum. Neither the System V nor BSD sum algorithm was selected. Since each of these was different and each was the default behavior on those systems, no realistic compromise was available if either were se? lected?some set of historical applications would break. Therefore, the name was changed to cksum. Although the historical sum commands will probably continue to be provided for many years, programs designed for portability across systems should use the new name.

The algorithm selected is based on that used by the ISO/IEC 8802?3:1996 standard (Ethernet) for the frame check sequence field. The algorithm used does not match the technical definition of a checksum; the term is used for historical reasons. The length of the file is included in the CRC calculation because this parallels inclusion of a length field by Ethernet in its CRC, but also because it guards against inadvertent collisions between files that begin with different series of zero octets. The chance that two different files produce identical CRCs is much greater when their lengths are not considered. Keeping the length and the checksum of the file itself separate would yield a slightly more robust algorithm, but historical usage has always been that a sin? gle number (the checksum as printed) represents the signature of the file. It was decided that historical usage was the more important con? sideration.

Early proposals contained modifications to the Ethernet algorithm that involved extracting table values whenever an intermediate result became zero. This was demonstrated to be less robust than the current method and mathematically difficult to describe or justify.

The calculation used is identical to that given in pseudo-code in the referenced Sarwate article. The pseudo-code rendition is:

```
X <- 0; Y <- 0;
for i <- m -1 step -1 until 0 do
    begin
    T <- X(1) ^ A[i];
    X(1) <- X(0); X(0) <- Y(1); Y(1) <- Y(0); Y(0) <- 0;
    comment: f[T] and f'[T] denote the T-th words in the
        table f and f' ;
    X <- X ^ f[T]; Y <- Y ^ f'[T];
    end
```

The pseudo-code is reproduced exactly as given; however, note that in the case of cksum, A[i] represents a byte of the file, the words X and Y are treated as a single 32-bit value, and the tables f and f' are a single table containing 32-bit values.

The referenced Sarwate article also discusses generating the table.

FUTURE DIRECTIONS

None.

SEE ALSO

The Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables

COPYRIGHT