NAME

md5, **sha1**, **sha224**, **sha256**, **sha384**, **sha512**, **sha512t224**, **sha512t256**, **rmd160**, **skein256**, **skein512**, **skein1024**, **md5sum**, **sha1sum**, **sha224sum**, **sha256sum**, **sha384sum**, **sha512sum**, **sha512t224sum**, **sha512t256sum**, **rmd160sum**, **skein256sum**, **skein512sum**, **skein1024sum**, **shasum** - calculate a message-digest fingerprint (checksum) for a file

SYNOPSIS

md5 [**-pqrtx**] [**-c** *string*] [**-s** *string*] [*file* ...]

md5sum [**-bctwz**] [**--binary**] [**--check**] [**--help**] [**--ignore-missing**] [**--quiet**] [**--status**] [**--strict**] [**--tag**] [**--text**] [**--version**] [**--warn**] [**--zero**] [*file* ...]

(All other hashes have the same options and usage.)

shasum [**-0bchqstUvw**] [**--01**] [**-a** | **--algorithm** *alg*] [**--binary**] [**--check**] [**--help**] [**--ignore-missing**] [**--quiet**] [**--status**] [**--strict**] [**--tag**] [**--text**] [**--UNIVERSAL**] [**--version**] [**--warn**] [*file* ...]

DESCRIPTION

The **md5**, **sha1**, **sha224**, **sha256**, **sha384**, **sha512**, **sha512t224**, **sha512t256**, **rmd160**, **skein256**, **skein512**, and **skein1024** utilities take as input a message of arbitrary length and produce as output a "fingerprint" or "message digest" of the input.

The **md5sum**, **sha1sum**, **sha224sum**, **sha256sum**, **sha384sum**, **sha512sum**, **sha512t224sum**, **sha512t256sum**, **rmd160sum**, **skein256sum**, **skein512sum**, and **skein1024sum** utilities do the same, but with command-line options and an output format that match those of their similary named GNU utilities.

The **shasum** utility does the same, but with command-line options and an output format that match those of the similarly named utility that ships with Perl.

It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The SHA-224 , SHA-256 , SHA-384 , SHA-512, RIPEMD-160, and SKEIN algorithms are intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

The MD5 and SHA-1 algorithms have been proven to be vulnerable to practical collision attacks and should not be relied upon to produce unique outputs, *nor should they be used as part of a cryptographic signature scheme.* As of 2017-03-02, there is no publicly known method to *reverse* either algorithm, i.e., to find an input that produces a specific output.

SHA-512t256 is a version of SHA-512 truncated to only 256 bits. On 64-bit hardware, this algorithm is approximately 50% faster than SHA-256 but with the same level of security. The hashes are not interchangeable.

SHA-512t224 is identical to SHA-512t256, but with the digest truncated to 224 bits.

It is recommended that all new applications use SHA-512 or SKEIN-512 instead of one of the other hash functions.

BSD OPTIONS

The following options are available in BSD mode, i.e. when the program is invoked with a name that does not end in "sum":

-c *string*, **--check=***string*

Compare the digest of the file against this string. If combined with the **-q** or **--quiet** option, the calculated digest is printed in addition to the exit status being set. (Note that this option is not yet useful if multiple files are specified.)

-p, **--passthrough**

Echo stdin to stdout and append the checksum to stdout.

-q, **--quiet**

Quiet mode -- only the checksum is printed out. Overrides the **-r** or **--reverse** option.

-r, **--reverse**

Reverses the format of the output. This helps with visual diffs. Does nothing when combined with the **-ptx** options.

-s *string*, **--string=***string*

Print a checksum of the given *string*.

-t, **--time-trial**

Run a built-in time trial. For the **-sum** versions, this is a nop for compatibility with coreutils.

-x, **--self-test**

Run a built-in test script.

GNU OPTIONS

The following options are available in GNU mode, i.e. when the program is invoked with a name that ends in "sum":

-b, **--binary**
Read files in binary mode.

-c, **--check**
The file passed as arguments must contain digest lines generated by the same digest algorithm in either classical BSD format or in GNU coreutils format. A line with the file name followed by a colon ":" and either OK or FAILED is written for each well-formed line in the digest file. If applicable, the number of failed comparisons and the number of lines that were skipped since they were not well-formed are printed at the end. The **--quiet** option can be used to quiesce the output unless there are mismatched entries in the digest.

--help Print a usage message and exit.

--ignore-missing
When verifying checksums, ignore files for which checksums are given but which aren’t found on disk.

--quiet
When verifying checksums, do not print anything unless the verification fails.

--status
When verifying checksums, do not print anything at all. The exit code will reflect whether verification succeeded.

--strict
When verifying checksums, fail if the input is malformed.

--tag Produce BSD-style output.

-t, **--text**
Read files in text mode. This is the default. Note that this implementation does not differentiate between binary and text mode.

--version
Print version information and exit.

-w, **--warn**
When verifying checksums, warn about malformed input.

-z, **--zero**

Terminate output lines with NUL rather than with newline.

PERL OPTIONS

The following options are available in Perl mode, i.e. when the program is invoked with the name "shasum":

-0, **--01**

Read files in bits mode: ASCII ‘0’ and ‘1’ characters correspond to 0 and 1 bits, respectively, and all other characters are ignored. See *BUGS*.

-a *alg*, **--algorithm** *alg*

Use the specified algorithm: "1" for SHA-1 (default), "xxx" for *xxx*-bit SHA-2 (e.g. "256" for SHA-256) or "xxxyyy" for *xxx*-bit SHA-2 truncated to *yyy* bits (e.g. "512224" for SHA-512/224).

-b, **--binary**

Read files in binary mode.

-c, **--check**

The file passed as arguments must contain digest lines generated by the same digest algorithm in either classical BSD format or in GNU coreutils format. A line with the file name followed by a colon ":" and either OK or FAILED is written for each well-formed line in the digest file. If applicable, the number of failed comparisons and the number of lines that were skipped since they were not well-formed are printed at the end. The **--quiet** option can be used to quiesce the output unless there are mismatched entries in the digest.

--help Print a usage message and exit.

--ignore-missing

When verifying checksums, ignore files for which checksums are given but which aren’t found on disk.

--quiet

When verifying checksums, do not print anything unless the verification fails.

--status

When verifying checksums, do not print anything at all. The exit code will reflect whether verification succeeded.

--strict

When verifying checksums, fail if the input is malformed.

--tag Produce BSD-style output.

-t, **--text**
Read files in text mode. This is the default. Note that this implementation does not differentiate between binary and text mode.

-U, **--UNIVERSAL**
Read files in universal mode: any CR-LF pair, as well as any CR not followed by LF, is translated to LF before the digest is computed.

--version
Print version information and exit.

-w, **--warn**
When verifying checksums, warn about malformed input.

EXIT STATUS

The **md5**, **sha1**, **sha224**, **sha256**, **sha384**, **sha512**, **sha512t224**, **sha512t256**, **rmd160**, **skein256**, **skein512**, and **skein1024** utilities exit 0 on success, 1 if at least one of the input files could not be read, and 2 if at least one file does not have the same hash as the **-c** option.

The **md5sum**, **sha1sum**, **sha224sum**, **sha256sum**, **sha384sum**, **sha512sum**, **sha512t224sum**, **sha512t256sum**, **rmd160**, **skein256**, **skein512**, **skein1024** and **shasum** utilities exit 0 on success and 1 if at least one of the input files could not be read or, when verifying checksums, does not have the expected checksum.

EXAMPLES

Calculate the MD5 checksum of the string "Hello".

```
$ md5 -s Hello
MD5 ("Hello") = 8b1a9953c4611296a827abf8c47804d7
```

Same as above, but note the absence of the newline character in the input string:

```
$ echo -n Hello | md5
8b1a9953c4611296a827abf8c47804d7
```

Calculate the checksum of multiple files reversing the output:

```
$ md5 -r /boot/loader.conf /etc/rc.conf
ada5f60f23af88ff95b8091d6d67bef6 /boot/loader.conf
d80bf36c332dc0fdc479366ec3fa44cd /etc/rc.conf
```

This is almost but not quite identical to the output from GNU mode:

```
$ md5sum /boot/loader.conf /etc/rc.conf
ada5f60f23af88ff95b8091d6d67bef6  /boot/loader.conf
d80bf36c332dc0fdc479366ec3fa44cd  /etc/rc.conf
```

Note the two spaces between hash and file name. If binary mode is requested, they are instead separated by a space and an asterisk:

```
$ md5sum -b /boot/loader.conf /etc/rc.conf
ada5f60f23af88ff95b8091d6d67bef6 */boot/loader.conf
d80bf36c332dc0fdc479366ec3fa44cd */etc/rc.conf
```

Write the digest for */boot/loader.conf* in a file named *digest*. Then calculate the checksum again and validate it against the checksum string extracted from the *digest* file:

```
$ md5 /boot/loader.conf > digest && md5 -c $(cut -f2 -d= digest) /boot/loader.conf
MD5 (/boot/loader.conf) = ada5f60f23af88ff95b8091d6d67bef6
```

Same as above but comparing the digest against an invalid string ("randomstring"), which results in a failure.

```
$ md5 -c randomstring /boot/loader.conf
MD5 (/boot/loader.conf) = ada5f60f23af88ff95b8091d6d67bef6 [ Failed ]
```

In GNU mode, the **-c** option does not compare against a hash string passed as parameter. Instead, it expects a digest file, as created under the name *digest* for */boot/loader.conf* in the example above.

```
$ md5 -c digest /boot/loader.conf
/boot/loader.conf: OK
```

The digest file may contain any number of lines in the format generated in either BSD or GNU mode. If a hash value does not match the file, "FAILED" is printed instead of "OK".

SEE ALSO

cksum(1), md5(3), ripemd(3), sha(3), sha256(3), sha384(3), sha512(3), skein(3)

R. Rivest, *The MD5 Message-Digest Algorithm*, RFC1321.

J. Burrows, *The Secure Hash Standard*, FIPS PUB 180-2.

D. Eastlake and P. Jones, *US Secure Hash Algorithm 1*, RFC 3174.

RIPEMD-160 is part of the ISO draft standard "ISO/IEC DIS 10118-3" on dedicated hash functions.

Secure Hash Standard (SHS): *https://www.nist.gov/publications/secure-hash-standard-shs*

The RIPEMD-160 page: *https://homes.esat.kuleuven.be/~bosselae/ripemd160.html*

BUGS

In bits mode, the original **shasum** script is capable of processing inputs of arbitrary length. This implementation is not, and will issue an error if the input length is not a multiple of eight bits.

ACKNOWLEDGMENTS

This utility was originally derived from a program which was placed in the public domain for free general use by RSA Data Security.

Support for SHA-1 and RIPEMD-160 was added by Oliver Eikemeier <*eik@FreeBSD.org*>.

Support for SHA-2 was added by Colin Percival <*cperciva@FreeBSD.org*> and Allan Jude <*allanjude@FreeBSD.org*>.

Support for SKEIN was added by Allan Jude <*allanjude@FreeBSD.org*>.

Compatibility with GNU coreutils was added by Warner Losh <*imp@FreeBSD.org*> and much expanded by Dag-Erling Smørgrav <*des@FreeBSD.org*>, who also added Perl compatibility.