

**NAME**

**RIPEMD160\_Init**, **RIPEMD160\_Update**, **RIPEMD160\_Final**, **RIPEMD160\_End**, **RIPEMD160\_File**, **RIPEMD160\_FileChunk**, **RIPEMD160\_Data** - calculate the RIPEMD160 message digest

**LIBRARY**

Message Digest (MD4, MD5, etc.) Support Library (libmd, -lmd)

**SYNOPSIS**

**#include** <sys/types.h>

**#include** <ripemd.h>

*void*

**RIPEMD160\_Init**(*RIPEMD160\_CTX \*context*);

*void*

**RIPEMD160\_Update**(*RIPEMD160\_CTX \*context, const unsigned char \*data, unsigned int len*);

*void*

**RIPEMD160\_Final**(*unsigned char digest[20], RIPEMD160\_CTX \*context*);

*char \**

**RIPEMD160\_End**(*RIPEMD160\_CTX \*context, char \*buf*);

*char \**

**RIPEMD160\_File**(*const char \*filename, char \*buf*);

*char \**

**RIPEMD160\_FileChunk**(*const char \*filename, char \*buf, off\_t offset, off\_t length*);

*char \**

**RIPEMD160\_Data**(*const unsigned char \*data, unsigned int len, char \*buf*);

**DESCRIPTION**

The **RIPEMD160\_** functions calculate a 160-bit cryptographic checksum (digest) for any number of input bytes. A cryptographic checksum is a one-way hash function; that is, it is computationally impractical to find the input corresponding to a particular output. This net result is a "fingerprint" of the input-data, which does not disclose the actual input.

The **RIPEMD160\_Init()**, **RIPEMD160\_Update()**, and **RIPEMD160\_Final()** functions are the core functions. Allocate an *RIPEMD160\_CTX*, initialize it with **RIPEMD160\_Init()**, run over the data with

**RIPEMD160\_Update()**, and finally extract the result using **RIPEMD160\_Final()**, which will also erase the *RIPEMD160\_CTX*.

The **RIPEMD160\_End()** function is a wrapper for **RIPEMD160\_Final()** which converts the return value to a 41-character (including the terminating `'\0'`) ASCII string which represents the 160 bits in hexadecimal.

The **RIPEMD160\_File()** function calculates the digest of a file, and uses **RIPEMD160\_End()** to return the result. If the file cannot be opened, a null pointer is returned. The **RIPEMD160\_FileChunk()** function is similar to **RIPEMD160\_File()**, but it only calculates the digest over a byte-range of the file specified, starting at *offset* and spanning *length* bytes. If the *length* parameter is specified as 0, or more than the length of the remaining part of the file, **RIPEMD160\_FileChunk()** calculates the digest from *offset* to the end of file. The **RIPEMD160\_Data()** function calculates the digest of a chunk of data in memory, and uses **RIPEMD160\_End()** to return the result.

When using **RIPEMD160\_End()**, **RIPEMD160\_File()**, or **RIPEMD160\_Data()**, the *buf* argument can be a null pointer, in which case the returned string is allocated with `malloc(3)` and subsequently must be explicitly deallocated using `free(3)` after use. If the *buf* argument is non-null it must point to at least 41 characters of buffer space.

## ERRORS

The **RIPEMD160\_End()** function called with a null *buf* argument may fail and return NULL if:

[ENOMEM]            Insufficient storage space is available.

The **RIPEMD160\_File()** and **RIPEMD160\_FileChunk()** may return NULL when underlying `open(2)`, `fstat(2)`, `lseek(2)`, or **RIPEMD160\_End(3)** fail.

## SEE ALSO

`md4(3)`, `md5(3)`, `sha(3)`, `sha256(3)`, `sha512(3)`, `skein(3)`

## HISTORY

These functions appeared in FreeBSD 4.0.

## AUTHORS

The core hash routines were implemented by Eric Young based on the published RIPEMD160 specification.

## BUGS

No method is known to exist which finds two files having the same hash value, nor to find a file with a

specific hash value. There is on the other hand no guarantee that such a method does not exist.