

NAME

SHA224_Init, **SHA224_Update**, **SHA224_Final**, **SHA224_End**, **SHA224_File**, **SHA224_FileChunk**, **SHA224_Data**, **SHA256_Init**, **SHA256_Update**, **SHA256_Final**, **SHA256_End**, **SHA256_File**, **SHA256_FileChunk**, **SHA256_Data** - calculate the FIPS 180-2 ‘SHA-256’ (or SHA-224) message digest

LIBRARY

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

SYNOPSIS

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

```
#include <sha224.h>
```

void

```
SHA224_Init(SHA224_CTX *context);
```

void

```
SHA224_Update(SHA224_CTX *context, const unsigned char *data, size_t len);
```

void

```
SHA224_Final(unsigned char digest[32], SHA224_CTX *context);
```

*char **

```
SHA224_End(SHA224_CTX *context, char *buf);
```

*char **

```
SHA224_File(const char *filename, char *buf);
```

*char **

```
SHA224_FileChunk(const char *filename, char *buf, off_t offset, off_t length);
```

*char **

```
SHA224_Data(const unsigned char *data, unsigned int len, char *buf);
```

```
#include <sha256.h>
```

void

```
SHA256_Init(SHA256_CTX *context);
```

void

SHA256_Update(*SHA256_CTX* *context, const unsigned char *data, size_t len);

void

SHA256_Final(unsigned char digest[32], *SHA256_CTX* *context);

char *

SHA256_End(*SHA256_CTX* *context, char *buf);

char *

SHA256_File(const char *filename, char *buf);

char *

SHA256_FileChunk(const char *filename, char *buf, off_t offset, off_t length);

char *

SHA256_Data(const unsigned char *data, unsigned int len, char *buf);

DESCRIPTION

The **SHA256_** functions calculate a 256-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 **SHA256_Init()**, **SHA256_Update()**, and **SHA256_Final()** functions are the core functions. Allocate an *SHA256_CTX*, initialize it with **SHA256_Init()**, run over the data with **SHA256_Update()**, and finally extract the result using **SHA256_Final()**, which will also erase the *SHA256_CTX*.

SHA256_End() is a wrapper for **SHA256_Final()** which converts the return value to a 65-character (including the terminating `'\0'`) ASCII string which represents the 256 bits in hexadecimal.

SHA256_File() calculates the digest of a file, and uses **SHA256_End()** to return the result. If the file cannot be opened, a null pointer is returned. **SHA256_FileChunk()** is similar to **SHA256_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, **SHA256_FileChunk()** calculates the digest from *offset* to the end of file. **SHA256_Data()** calculates the digest of a chunk of data in memory, and uses **SHA256_End()** to return the result.

When using **SHA256_End()**, **SHA256_File()**, or **SHA256_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 65 characters

of buffer space.

SHA224 is identical SHA256, except it has slightly different initialization vectors, and is truncated to a shorter digest.

ERRORS

The **SHA256_End()** function called with a null buf argument may fail and return NULL if:

[ENOMEM] Insufficient storage space is available.

The **SHA256_File()** and **SHA256_FileChunk()** may return NULL when underlying `open(2)`, `fstat(2)`, `lseek(2)`, or `SHA256_End(3)` fail.

SEE ALSO

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

HISTORY

These functions appeared in FreeBSD 6.0.

AUTHORS

The core hash routines were implemented by Colin Percival based on the published FIPS 180-2 standard.

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.