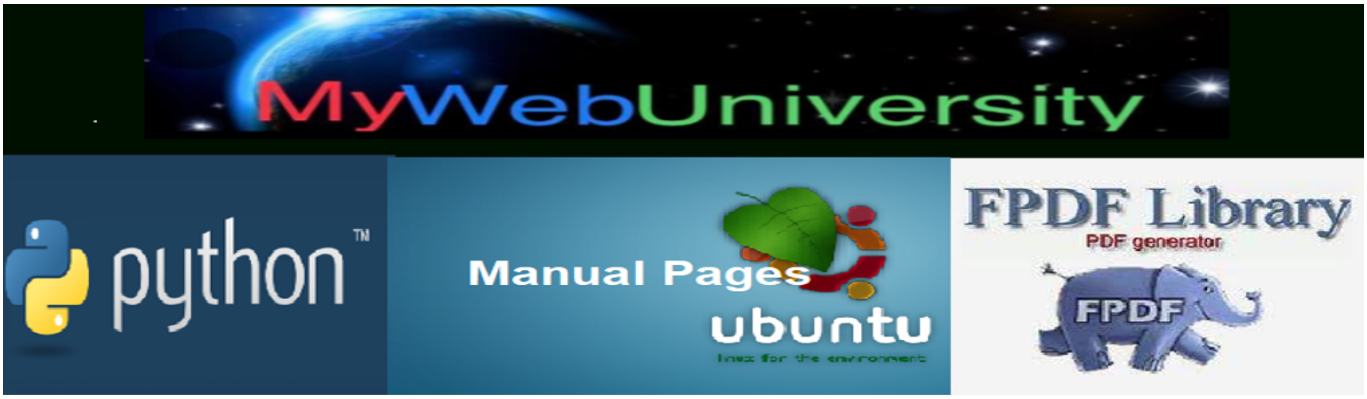

Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::Digest::SHA3_384.3pm'

$ man Crypt::Digest::SHA3_384.3pm

NAME

Crypt::Digest::SHA3_384 - Hash function SHA3-384 [size: 384 bits]

SYNOPSIS

```
### Functional interface:
use Crypt::Digest::SHA3_384 qw( sha3_384 sha3_384_hex sha3_384_b64 sha3_384_b64u
                                sha3_384_file sha3_384_file_hex sha3_384_file_b64 sha3_384_file_b64u );

# calculate digest from string/buffer
$sha3_384_raw  = sha3_384('data string');
$sha3_384_hex  = sha3_384_hex('data string');
$sha3_384_b64  = sha3_384_b64('data string');
$sha3_384_b64u = sha3_384_b64u('data string');
# calculate digest from file
$sha3_384_raw  = sha3_384_file('filename.dat');
$sha3_384_hex  = sha3_384_file_hex('filename.dat');
$sha3_384_b64  = sha3_384_file_b64('filename.dat');
$sha3_384_b64u = sha3_384_file_b64u('filename.dat');
# calculate digest from filehandle
$sha3_384_raw  = sha3_384_file(*FILEHANDLE);
$sha3_384_hex  = sha3_384_file_hex(*FILEHANDLE);
```

```
$sha3_384_b64  = sha3_384_file_b64(*FILEHANDLE);
$sha3_384_b64u = sha3_384_file_b64u(*FILEHANDLE);

### OO interface:
use Crypt::Digest::SHA3_384;

$d = Crypt::Digest::SHA3_384->new;
$d->add('any data');
$d->addfile('filename.dat');
$d->addfile(*FILEHANDLE);
$result_raw  = $d->digest;     # raw bytes
$result_hex  = $d->hexdigest;  # hexadecimal form
$result_b64  = $d->b64digest;  # Base64 form
$result_b64u = $d->b64udigest; # Base64 URL Safe form
```

DESCRIPTION

Provides an interface to the SHA3_384 digest algorithm.

EXPORT

Nothing is exported by default.

You can export selected functions:

```
use Crypt::Digest::SHA3_384 qw(sha3_384 sha3_384_hex sha3_384_b64 sha3_384_b64u
                               sha3_384_file sha3_384_file_hex sha3_384_file_b64 sha3_384_file_b64u);
```

Or all of them at once:

```
use Crypt::Digest::SHA3_384 ':all';
```

FUNCTIONS

sha3_384

Logically joins all arguments into a single string, and returns its SHA3_384 digest

encoded as a binary string.

```
$sha3_384_raw = sha3_384('data string');
#or
$sha3_384_raw = sha3_384('any data', 'more data', 'even more data');
```

sha3_384_hex

Logically joins all arguments into a single string, and returns its SHA3_384 digest encoded as a hexadecimal string.

```
$sha3_384_hex = sha3_384_hex('data string');
#or
$sha3_384_hex = sha3_384_hex('any data', 'more data', 'even more data');
```

sha3_384_b64

Logically joins all arguments into a single string, and returns its SHA3_384 digest encoded as a Base64 string, with trailing '=' padding.

```
$sha3_384_b64 = sha3_384_b64('data string');
#or
$sha3_384_b64 = sha3_384_b64('any data', 'more data', 'even more data');
```

sha3_384_b64u

Logically joins all arguments into a single string, and returns its SHA3_384 digest encoded as a Base64 URL Safe string (see RFC 4648 section 5).

```
$sha3_384_b64url = sha3_384_b64u('data string');
#or
$sha3_384_b64url = sha3_384_b64u('any data', 'more data', 'even more data');
```

sha3_384_file

Reads file (defined by filename or filehandle) content, and returns its SHA3_384 digest encoded as a binary string.

```
$sha3_384_raw = sha3_384_file('filename.dat');
#or
$sha3_384_raw = sha3_384_file(*FILEHANDLE);
```

sha3_384_file_hex

Reads file (defined by filename or filehandle) content, and returns its SHA3_384 digest encoded as a hexadecimal string.

```
$sha3_384_hex = sha3_384_file_hex('filename.dat');
#or
$sha3_384_hex = sha3_384_file_hex(*FILEHANDLE);
```

BEWARE: You have to make sure that the filehandle is in binary mode before you pass it as argument to the addfile() method.

sha3_384_file_b64

Reads file (defined by filename or filehandle) content, and returns its SHA3_384 digest encoded as a Base64 string, with trailing '=' padding.

```
$sha3_384_b64 = sha3_384_file_b64('filename.dat');
#or
$sha3_384_b64 = sha3_384_file_b64(*FILEHANDLE);
```

sha3_384_file_b64u

Reads file (defined by filename or filehandle) content, and returns its SHA3_384 digest encoded as a Base64 URL Safe string (see RFC 4648 section 5).

```
$sha3_384_b64url = sha3_384_file_b64u('filename.dat');
#or
$sha3_384_b64url = sha3_384_file_b64u(*FILEHANDLE);
```

The OO interface provides the same set of functions as Crypt::Digest.

new

```
$d = Crypt::Digest::SHA3_384->new();
```

clone

```
$d->clone();
```

reset

```
$d->reset();
```

add

```
$d->add('any data');
#or
$d->add('any data', 'more data', 'even more data');
```

addfile

```
$d->addfile('filename.dat');
#or
$d->addfile(*FILEHANDLE);
```

add_bits

```
$d->add_bits($bit_string);   # e.g. $d->add_bits("111100001010");
#or
$d->add_bits($data, $nbits); # e.g. $d->add_bits("\xF0\xA0", 16);
```

hashsize

```
$d->hashsize;
#or
Crypt::Digest::SHA3_384->hashsize();
#or
Crypt::Digest::SHA3_384::hashsize();
```

digest

```
$result_raw = $d->digest();
```

hexdigest

```
$result_hex = $d->hexdigest();
```

b64digest

```
$result_b64 = $d->b64digest();
```

b64udigest

```
$result_b64url = $d->b64udigest();
```

SEE ALSO

- CryptX, Crypt::Digest
- <https://en.wikipedia.org/wiki/SHA-3>