NAME

rtw_short_hash, rtw_long_hash, rtw_longlong_hash, strhash − compute hash of a sequence of bytes

SYNOPSIS

#include "mjsu.h"

ULONG rtw_long_hash(BYTE *key, size_t len);

USHORT rtw_short_hash(BYTE *key, size_t len);

ULLONG rtw_longlong_hash(BYTE *key, size_t len);

USHORT strhash(CHAR *key);

DESCRIPTION

rtw_long_hash() computes a hash value of the input sequence of bytes at *key* whose length in bytes is given by *len*.

All the bits of the return value are significant, even on platforms where long integers consist of a truly strange number of bits (eg: 137). This allows the returned value to be scaled down to a desired power-of-two range by simply masking off sufficient high-order bits, without losing any spectral distribution in the output values for different inputs.

rtw_short_hash() performs the same function but produces a short hash value instead of a long.

rtw_longlong_hash() performs the same function but produces a "long long" hash value instead of a long; this function is only available with platforms/compilers that support the "long long" data-type.

strhash() is a convenience function that performs the equivalent of rtw_short_hash() on NUL-terminated strings.

RETURNS

These functions return a highly spectrally-distributed hash value. A given input always produces the same output value.

SEE ALSO

mjsu(7).

AVAILABILITY

The **rtw_long_hash()**, **rtw_short_hash()** and **strhash()** functions are written in C, conforming to ANSI X3.159-1989.

The **rtw_longlong_hash()** uses the C99 "**long long**" datatype, and is thus is only available where the C compiler concerned supports the "**long long**" datatype, and only when the symbol **HAVE_LONGLONG** is defined before the first #include directive that refers to mjsu.h.

NOTES

These functions do not generate "cryptographiclly secure" hash values, even though the outputs may appear inscrutable. The outputs are intended for traditional hashing purposes: randomly-indexed arrays, fast lookup-tables, and other situations where dealing with integer values in lieu of strings or variable-length indexes is easier or faster.