



## ***Linux Ubuntu 26.04 Manual Pages on command 'htole32.3'***

**\$ man htole32.3**

endian(3)                      Library Functions Manual                      endian(3)

### **NAME**

htobe16, htole16, be16toh, le16toh, htobe32, htole32, be32toh, le32toh, htobe64, htole64,  
be64toh, le64toh - convert values between host and big-/little-endian byte order

### **LIBRARY**

Standard C library (libc, -lc)

### **SYNOPSIS**

```
#include <endian.h>

uint16_t htobe16(uint16_t host_16bits);
uint16_t htole16(uint16_t host_16bits);
uint16_t be16toh(uint16_t big_endian_16bits);
uint16_t le16toh(uint16_t little_endian_16bits);
uint32_t htobe32(uint32_t host_32bits);
uint32_t htole32(uint32_t host_32bits);
uint32_t be32toh(uint32_t big_endian_32bits);
uint32_t le32toh(uint32_t little_endian_32bits);
uint64_t htobe64(uint64_t host_64bits);
uint64_t htole64(uint64_t host_64bits);
uint64_t be64toh(uint64_t big_endian_64bits);
uint64_t le64toh(uint64_t little_endian_64bits);
```

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

htobe16(), htole16(), be16toh(), le16toh(), htobe32(), htole32(), be32toh(), le32toh(),  
htobe64(), htole64(), be64toh(), le64toh():

Since glibc 2.19:

`_DEFAULT_SOURCE`

In glibc up to and including 2.19:

`_BSD_SOURCE`

## DESCRIPTION

These functions convert the byte encoding of integer values from the byte order that the current CPU (the "host") uses, to and from little-endian and big-endian byte order.

The number, `nn`, in the name of each function indicates the size of integer handled by the function, either 16, 32, or 64 bits.

The functions with names of the form "`htobenn`" convert from host byte order to big-endian order.

The functions with names of the form "`htolenn`" convert from host byte order to little-endian order.

The functions with names of the form "`benntoh`" convert from big-endian order to host byte order.

The functions with names of the form "`lenntoh`" convert from little-endian order to host byte order.

## VERSIONS

Similar functions are present on the BSDs, where the required header file is `<sys/endian.h>` instead of `<endian.h>`. Unfortunately, NetBSD, FreeBSD, and glibc haven't followed the original OpenBSD naming convention for these functions, whereby the `nn` component always appears at the end of the function name (thus, for example, in NetBSD, FreeBSD, and glibc, the equivalent of OpenBSDs "`betoh32`" is "`be32toh`").

## STANDARDS

None.

## HISTORY

glibc 2.9.

These functions are similar to the older `byteorder(3)` family of functions. For example, `be32toh()` is identical to `ntohl()`.

The advantage of the `byteorder(3)` functions is that they are standard functions available on all UNIX systems. On the other hand, the fact that they were designed for use in the context of TCP/IP means that they lack the 64-bit and little-endian variants described in this page.

## EXAMPLES

The program below display the results of converting an integer from host byte order to both little-endian and big-endian byte order. Since host byte order is either little-endian or big-endian, only one of these conversions will have an effect. When we run this program on a little-endian system such as x86-32, we see the following:

```
$ ./a.out;
x.u32 = 0x44332211
htole32(x.u32) = 0x44332211
htobe32(x.u32) = 0x11223344
```

### Program source

```
#include <endian.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

int
main(void)
{
    union {
        uint32_t u32;
        uint8_t arr[4];
    } x;

    x.arr[0] = 0x11; /* Lowest-address byte */
    x.arr[1] = 0x22;
    x.arr[2] = 0x33;
    x.arr[3] = 0x44; /* Highest-address byte */

    printf("x.u32 = %#x\n", x.u32);
    printf("htole32(x.u32) = %#x\n", htole32(x.u32));
    printf("htobe32(x.u32) = %#x\n", htobe32(x.u32));
    exit(EXIT_SUCCESS);
}
```

### SEE ALSO

[bswap\(3\)](#), [byteorder\(3\)](#)