`htons()`, `htonl()`, `ntohs()`, `ntohl()`

Convert multi-byte integer types from host byte order to network byte order

Prototypes

```
#include <netinet/in.h>

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

Description

Just to make you really unhappy, different computers use different byte orderings internally for their multibyte integers (i.e. any interger that's larger than a `char`.) The upshot of this is that if you `send()` a two-byte `short int` from an Intel box to a Mac (before they became Intel boxes, too, I mean), what one computer thinks is the number **1**, the other will think is the number **256**, and vice-versa.

The way to get around this problem is for everyone to put aside their differences and agree that Motorola and IBM had it right, and Intel did it the weird way, and so we all convert our byte orderings to "big-endian" before sending them out. Since Intel is a "little-endian" machine, it's far more politically correct to call our preferred byte ordering "Network Byte Order". So these functions convert from your native byte order to network byte order and back again.

(This means on Intel these functions swap all the bytes around, and on PowerPC they do nothing because the bytes are already in Network Byte Order. But you should always use them in your code anyway, since someone might want to build it on an Intel machine and still have things work properly.)

Note that the types involved are 32-bit (4 byte, probably `int`) and 16-bit (2 byte, very likely `short`) numbers. 64-bit machines might have a `htonll()` for 64-bit ints, but I've not seen it. You'll just have to write your own.

Anyway, the way these functions work is that you first decide if you're converting *from* host (your machine's) byte order or from network byte order. If "host", the the first letter of the function you're going to call is "h". Otherwise it's "n" for "network". The middle of the function name is always "to" because you're converting from one "to" another, and the penultimate letter shows what you're converting *to*. The last letter is the size of the data, "s" for short, or "l" for long. Thus:

`htons()` *h*ost *to* *n*etwork *s*hort

`htonl()` *h*ost *to* *n*etwork *l*ong

`ntohs()` *n*etwork *to* *h*ost *s*hort

`ntohl()` *n*etwork *to* *h*ost *l*ong

Return Value

Each function returns the converted value.

Example

```
uint32_t some_long = 10;
uint16_t some_short = 20;

uint32_t network_byte_order;

// convert and send
network_byte_order = htonl(some_long);
send(s, &network_byte_order, sizeof(uint32_t), 0);

some_short == ntohs(htons(some_short)); // this expression is true
```
