Network Byte Order Conversion

Contents

1   Network Byte Order Conversion

1.1   Introduction

This proposal adds support to C++ for converting between host and network byte order.

1.2   Motivation and Scope

Converting between host and network byte ordering is an essential component of many network programs. This proposal adds support for the four existing byte order converting function specified by IEEE Std 1003.1-2008 that have been in use since the 1970s. Additionally two generic functions, and their specializations for unsigned integer types, are specified for converting between network and host byte order.

1.3   Design Decisions

The functions `htonl()`, `htons()`, `ntohl()`, and `ntohs()` are intentionally included even though they are not necessary given the `hton<T>()` and `ntoh<T>()` template functions. This is to maintain compatibility with the POSIX standard (IEEE Std 1003.1-2008) and to allow backwards compatibility with existing networking programs. The function names are also chosen to match existing names rather than using a more verbose and descriptive name such as `host_to_network`. This choice is made as, within the network programming domain, `ntoh` and `hton` are established vocabulary terms.

`hton<T>()` and `ntoh<T>()` are specified as generic templates rather than listing overloads for `hton()` and `ntoh()`. This allows user extension of this function.

The current specification allows the four non-template functions to be macros. This does not present implementation concerns, is addressed by note #174 from paragraph 6 in section `17.6.1.2`, and is already dealt with for names in the `<cerrno>` header.

An alternate proposal is to standardize a more general byte order conversion library like Boost Endian. Such a proposal expands the scope beyond that which is immediately useful for network programming. The choice to provide a network domain specific byte order conversion facility allows a more immediately useful interface based on existing practice that does not in any way limit the future introduction of a more general purpose byte order conversion facility. Should such a facility be standardized in the future the implementation of this proposal would simply alias names from there. As stated before, within the network programming domain `hton` and `ntoh` are established vocabulary terms and are useful names within `net::` even if a more general facility is developed.

1.4   Technical Specifications

Header `<net>` Synopsis

```
namespace std {
namespace experimental {
namespace net {

constexpr uint32_t htonl(uint32_t host) noexcept;
constexpr uint16_t htons(uint16_t host) noexcept;
template <class T>
  constexpr T hton(T host) noexcept = delete;
template <>
  constexpr unsigned-integral hton(unsigned-integral host) noexcept;

constexpr uint32_t ntohl(uint32_t network) noexcept;
constexpr uint16_t ntohs(uint16_t network) noexcept;
```

```
template <class T>
  constexpr T ntoh(T network) noexcept = delete;
template <>
  constexpr unsigned-integral ntoh(unsigned-integral network) noexcept;

} // namespace net
} // namespace experimental
} // namespace std
```

[1] The `<net>` header is available if `uint8_t`, `uint16_t`, `uint32_t`, and `uint64_t` are provided by `<cstdint>`.

[2] For each unsigned integer type *unsigned-integral*, there shall be explicit specializations of the `hton()` and `ntoh()` templates.

[3] *Network byte order* is *big-endian*, or most significant byte first (RFC 2781 section 3.1). This byte order is used by certain network data formats as it passes through the network. Host byte order is the endianness of the host machine.

```
constexpr uint32_t htonl(uint32_t host) noexcept;
constexpr uint16_t htons(uint16_t host) noexcept;
template <>
  constexpr unsigned-integral  hton(unsigned-integral  host) noexcept;
```

[4]   *Returns:* The argument value converted from host to network byte order.

```
constexpr uint32_t ntohl(uint32_t network) noexcept;
constexpr uint16_t ntohs(uint16_t network) noexcept;
template <>
  constexpr unsigned-integral  ntoh(unsigned-integral  network) noexcept;
```

[5]   *Returns:* The argument value converted from network to host byte order.

1.5   Acknowledgments

Thanks for the editorial feedback from Darin Pantley and Rob Pratte. Thanks to Benjamin De Kosnik for early implementation validation. Thanks to Christof Meerwald for the suggestion of using `= delete`. Thanks to Frank Birbacher for the suggestion to use `noexcept`. Thanks to the rest of SG4 group in attendance in Bristol for good discussion and feedback.

1.6   References

— Open Group Base Specifications Issue 7, IEEE Std 1003.1-2008

— BYTEORDER(3) - BSD Library Functions Manual (4 June 1993)

— Boost Endian - http://boost.cowic.de/rc/endian/doc/