**EXPERIMENT NO-8**

***Aim:***To study and implement Host byte Order.

**Theory:**  Host Byte Order refers to how bytes are arranged when referring to the computer architecture of a host computing platform. Due to pervasiveness of the Intel architecture, this is generally Little Endian This means least significant byte in the smallest address in a word.

Here we can simply comprehend big endian and little endian as Network Byte Order and Host Byte Order respectively. What is the difference between both two?

Usually in our host computer we store the data byte as Host Byte Order, for example, we store a integer in the RAM which might occupies 4 Byte, as Host Byte Order the higher Byte would be stored at the lower address of RAM, and the lower Byte would be stored at the higher address of RAM. However, contrast to this, Network Byte Order just take the totally opposite way to store the data, says, it will store the lower Byte at the lower address, and the higher Byte will stay at higher address.

For the current communication of network, we normally exchange the information by surveying the data package, every two host wants to communicate with each other must send and receive data package through network. In order to maintain the identity of data through the transmission in the network, the order of the Byte storage must changed before sending and after receiving the data.

Unfortunately, not all computers store the bytes that comprise a multibyte value in the same order. Consider a 16-bit internet that is made up of 2 bytes. There are two ways to store this value.

* **Little Endian** − In this scheme, low-order byte is stored on the starting address (A) and high-order byte is stored on the next address (A + 1).
* **Big Endian** − In this scheme, high-order byte is stored on the starting address (A) and low-order byte is stored on the next address (A + 1).

To allow machines with different byte order conventions communicate with each other, the Internet protocols specify a canonical byte order convention for data transmitted over the network. This is known as Network Byte Order.

While establishing an Internet socket connection, you must make sure that the data in the sin\_port and sin\_addr members of the sockaddr\_in structure are represented in Network Byte Order.

Network byte order is decided to be big-endian in all cases, and the host byte order depends on host's platform. The network byte order is defined to always be big-endian, which may differ from the host byte order on a particular machine. Using network byte ordering for data exchanged between hosts allows hosts using different architectures to exchange address information without confusion because of byte ordering.

## **Byte Ordering Functions**

Routines for converting data between a host's internal representation and Network Byte Order are as follows −

|  |  |
| --- | --- |
| **Function** | **Description** |
| htons() | Host to Network Short |
| htonl() | Host to Network Long |
| ntohl() | Network to Host Long |
| ntohs() | Network to Host Short |

Listed below are some more detail about these functions −

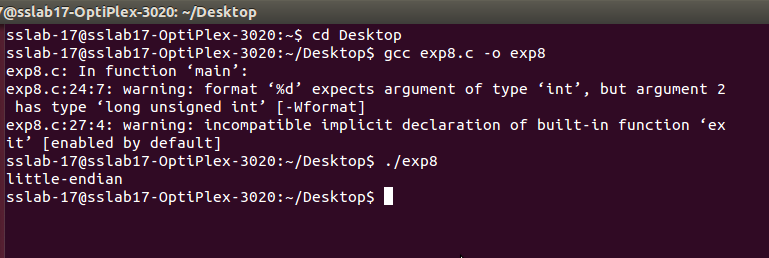
* **unsigned short htons(unsigned short hostshort)** − This function converts 16-bit (2-byte) quantities from host byte order to network byte order.
* **unsigned long htonl(unsigned long host**
* **long)** − This function converts 32-bit (4-byte) quantities from host byte order to network byte order.
* **unsigned short ntohs(unsigned short netshort)** − This function converts 16-bit (2-byte) quantities from network byte order to host byte order.
* **unsigned long ntohl(unsigned long netlong)** − This function converts 32-bit quantities from network byte order to host byte order.

These functions are macros and result in the insertion of conversion source code into the calling program. On little-endian machines, the code will change the values around to network byte order. On big-endian machines, no code is inserted since none is needed; the functions are defined as null.

## **Program to Determine Host Byte Order**

#include <stdio.h>  
int main(int argc, char \*\*argv) {  
 union {  
 short s;  
 char c[sizeof(short)];  
 }un;  
 un.s = 0x0102;  
   
 if (sizeof(short) == 2) {  
 if (un.c[0] == 1 && un.c[1] == 2)  
 printf("big-endian\n");  
   
 else if (un.c[0] == 2 && un.c[1] == 1)  
 printf("little-endian\n");  
   
 else  
 printf("unknown\n");  
 }  
 else {  
 printf("sizeof(short) = %d\n", sizeof(short));  
 }  
   
 exit(0);  
}

***OUTPUT***



**Conclusion:** Thus, we studied about the types , functions, program and method of conversion of Host Byte Order.