Socket Programming in UNIX

Prof. Anand D Dave
Department of Information Technology,
Dharmsinh Desai University, Nadiad.

Distributed Computing: Client- Server Socket Address Structure

- a pointer to a socket address structure as an argument in most socket functions.
- Each supported protocol suite defines its own socket address structure.
- The names of these structures begin with sockaddr_ and end with a unique suffix for each protocol suite

Distributed Computing: Client- Server Socket Address Structure

- Generic Socket address structure.

  Type of argument?

  – With ANSI C, solution is simple. Use void*

  – Socket functions exists before ANSI C. Solution chosen in 1982 was to define a generic socket address structure.

sockaddr (Page 60, Stevens Vol.1)

```
#include <sys/socket.h>
        struct sockaddr {
                uint8_t sa_len;
                sa_family_t sa_family; /* (unsigned short) address family,
                                          AF_xxx or PF_xxx*/
                char sa_data[14]; /* 14 bytes of protocol-specific address*/
                };
```

Distributed Computing: Client- Server Socket Address Structure

- IPv4 socket address structure
  - commonly called an "Internet socket address structure,
  - It is named sockaddr_in
  - defined in the <netinet/in.h> header

in_addr (Page 58, Stevens Vol.1)

```
#include <netinet/in.h>
    struct in_addr {
            in_addr_t s_addr; /* that's a 32-bit int (4 bytes)
                                                    (uint32_t) */
                    };
```

Distributed Computing: Client- Server Socket Address Structure

In our program the following structure is going to be used

sockaddr_in (Page 58, Stevens Vol.1)

```
#include <netinet/in.h>
   struct sockaddr_in {
  uint8_t sin_len;
  sa_family_t sin_family; /* (unsigned short) Address
                                        family, AF_INET */
  in_port_t sin_port; /* (unsigned short int) Port number */
  struct in_addr sin_addr; /* Internet address unsigned char*/
 sin_zero[8]; /* Same size as struct sockaddr*/
};
```

Distributed Computing: Client- Server Socket Address Structure

- Both the IPv4 address and the TCP or UDP port number are always stored in the structure in network byte order.
- The 32-bit IPv4 address can be accessed in two different ways: – For example,

struct sockaddr_in serv;

(1) serv.sin_addr references the 32-bit IPv4 address as an in_addr structure,

(2) while serv.sin_addr.s_addr references the same 32-bit IPv4 address as an in_addr_t (typically an unsigned 32-bit integer).

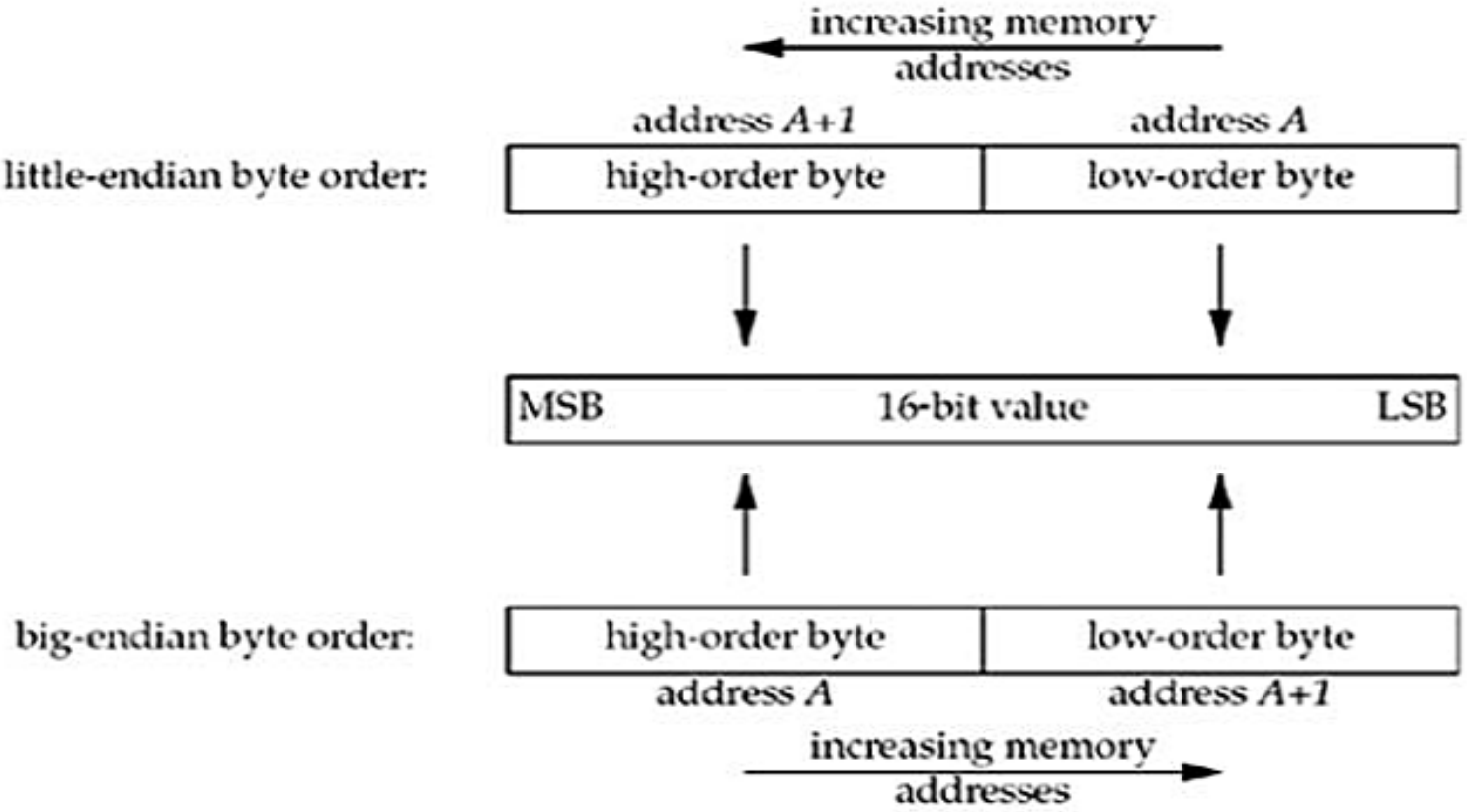
• The sin_zero member is unused,

– we always set it to 0 when filling in one of these structures.

– By convention, we always set the entire structure to 0 before filling it in, not just the sin_zero member.

Distributed Computing: Client- Server Byte Ordering functions

- There are two different formats (little endian and big endian). Any computer system uses one of this.(it is specific to hardware architecture).



Distributed Computing: Client- Server Byte Ordering functions

- Host byte order: byte order used by a given system is called host byte order.
- Network byte order : Network byte order used for network program by networking protocol.(E.g for TCP, kernel) is network byte order.
- We need conversion functions for short and long data types.

  #include<netinet/in.h>
- Conversion to network byte order. l for long(4 byte) and s for short (2 byte)

  uint16_t htons(uint16_t net16bitvalue);

  uint32_t htonl(uint32_t net32bitvalue);
- Conversion to Host byte order:

  uint16_t ntohs(uint16_t net16bitvalue);

  uint32_t ntohl(uint32_t net32bitvalue);

Distributed Computing: Client- Server
Byte Manipulation functions

- Byte Manipulation functions

#include<string.h>

void bzero(void *dest, size_t nbytes);

- We use this function to initialize address structure with 0 value in all its byte.

Distributed Computing: Client- Server Address conversion functions

- These functions convert Internet addresses between ASCII strings (what humans prefer to use- dotted -decimal string,e.g., "206.168.112.96" and network byte ordered binary values (values that are stored in socket address structures, 32 bytes values).
- Two groups of address conversion functions.

#include <arpa/inet.h>

int inet_aton(const char * strptr , struct in_addr * addrptr);

Returns: 1 if string was valid, 0 on error

char *inet_ntoa(struct in_addr inaddr);

Returns: pointer to dotted-decimal string

in_addr_t inet_addr(const char * strptr);

Returns: 32-bit binary network byte ordered IPv4 address; INADDR_NONE if error

Distributed Computing: Client- Server
Address conversion functions

- The newer functions, inet_pton and inet_ntop , handle both IPv4 and IPv6 addresses.

#include<arpa/inet.h>
int inet_pton(int family , const char * strptr , void * addrptr);
• Returns: 1 if OK, 0 if input not a valid presentation format, -1 on error
• The family argument for both functions is either AF_INET or AF_INET6 .

const char *inet_ntop(int family , const void * addrptr , char * strptr , size_t len);

Returns: pointer to result if OK, NULL on error.
• The len argument is the size of the destination, to prevent the function from overflowing the caller's buffer.

• The strptr argument to inet_ntop cannot be a null pointer. The ***caller must allocate memory for the destination*** and specify its size. On success, this pointer is the return value of the function. (See example.)

References

- Sources :
- https://www.javatpoint.com/osi-model
- https://www.tutorialspoint.com/software_architecture_design/distributed_architecture.htm
- https://sites.google.com/site/prajapatiharshadb/class-notes-for-students
- UNIX Network Programming by W. Richard Stevens, Prentice Hall Publication
- Distributed Computing: Concepts &Applications: by M. L. Liu Addison Wiselly