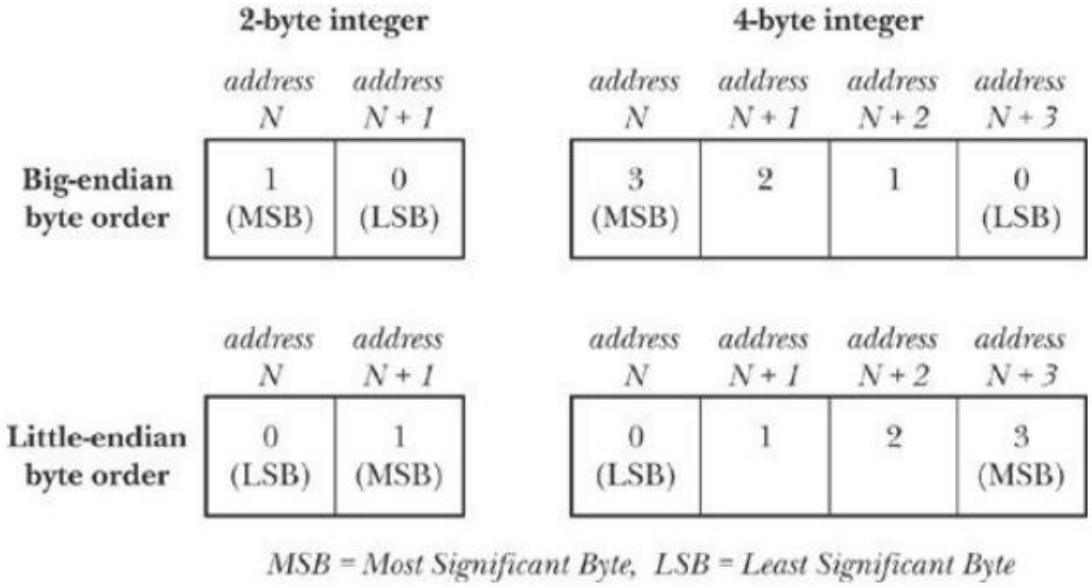
Լաբորատոր աշխատանք 8

Սոկետներ (Sockets)
Ինտերնետային դոմեյն

Ինտերնետային դոմեյնի հոսքային սոկետների իրականացման հիմքում ընկած է TCP-ն, իսկ դեյտագրամային սոկետների իրականացման հիմքում՝ UDP-ն:

IP հասցեները և պորտերի համարները ամբողջ արժեքներ են: Այս արժեքները ցանցով փոխանցելիս խնդիր կարող է հանդիսանալ այն, որ տարբեր ճարտարապետություններ պահում են ամբողջ թվերի բայթերը տարբեր հերթականությամբ: Այն ճարտարապետությունները, որոնք առաջին տեղում պահում են ավագ բայթը, կոչվում են **big endian**: Այն ճարտարապետությունները, որոնք առաջին տեղում պահում են կրտսեր բայթը, կոչվում են **little endian**:



Նկ. 1. Big-endian և little-endian բայթերի հաջորդականությունը

Քանի որ այս արժեքները պետք է փոխանցվեն ցանցով և հասկանալի լինեն բոլոր հոսթերին, ապա պետք է կիրառվի որևէ ստանդարտ: Այդ ստանդարտն անվանում են ցանցի բայթերի կարգ (network byte order), որը big-endian է: Ծրագրերում IP հասցեներ և պորտեր օգտագործելիս դրանք պետք է նախ բերել ստանդարտ տեսքի: Հոսթի և ցանցի միջև այս փոխակերպումները կատարելու համար օգտագործվում են htons(), htonl(), ntohs(), ntohl() ֆունկցիաները:

#include <arpa/inet.h>

uint16_t **htons**(uint16_t host_uint16); // host to network short

uint32_t **htonl**(uint32_t host_uint32); // host to network long

uint16_t **ntohs**(uint16_t net_uint16); // network to host short

uint32_t **ntohl**(uint32_t net_uint32); // network to host long

inet_pton() և inet_ntop() ֆունկցիաները

inet_pton() և inet_ntop() ֆունկցիաները ձևափոխում են ներկայացման ֆորմատի IP հասցեները երկուական ֆորմատի, և հակառակը: Ֆունկցիաների անվանումների մեջ p տառը նշանակում է "presentation", իսկ n տառը՝ "network": Presentation-ը մարդու կողմից ընթեռնելի տող է, ինչպիսին է օրինակ 204.152.189.116:

#include <arpa/inet.h>

int **inet_pton**(int **domain**, const char ***src_str**, void ***addrptr**);

Ձևափոխում է stc_str արգումենտի ներկայացման ֆորմատի տողը IP հասցեի: domain արգումենտը պետք է սահմանվի որպես AF_INET կամ AF_INET6: Ձևափոխված հասցեն տեղադրվում է addrptr ցուցիչի ստրուկտուրայում:

#include <arpa/inet.h>

const char ***inet_ntop**(int **domain**, const void ***addrptr**, char ***dst_str**, size_t **len**);

inet_ntop() ֆունկցիան իրականացնում է հակառակ գործառույթը:

IPv4 սոկետի հասցեն

IPv4 սոկետի հասցեն պահվում է **sockaddr_in** ստրուկտուրայում:

#include <netinet/in.h>

```
struct in_addr { /* IPv4 4-byte address */
        in_addr_t s_addr; /* Unsigned 32-bit integer */
};

struct sockaddr_in { /* IPv4 socket address */
        sa_family_t sin_family; /* Address family (AF_INET) */
        in_port_t sin_port; /* Port number */
        struct in_addr sin_addr; /* IPv4 address */
        unsigned char __pad[X]; /* Pad to size of 'sockaddr' structure (16 bytes) */

};
```

sin_port և **sin_addr** դաշտերը պահում են պորտի համարը և IP հասցեն՝ 2-ն էլ ցանցի բայթերի կարգով: **in_port_t** և **in_addr_t** տիպերը unsigned integer տիպեր են, համապատասխանաբար 16 և 32 բիթ երկարությամբ:

ipv4_dg_server, ipv4_dg_client ծրագրերը

Այս ծրագրերը ներկայացնում են INET դոմեյնում դեյտագրամային սոկետների աշխատանքը: Client ծրագիրը server-ին ուղարկում է STDIN հոսքից ստացված տվյալները, server-ը դրանք դարձնում է մեծատառ և վերադարձնում client-ին: Ստացված արդյունքն արտածվում է:

Client ծրագիրը որպես հրամանային տողի արգումենտ ընդունում է սերվերի IP հասցեն: Տվյալ օրինակում կիրառվում է 127.0.0.1 հասցեն, որը համարվում է IPv4 ցանցի հետադարձ հասցե (loopback address): Հետադարձ հասցեներ են համարվում նաև 127.0.0.0 / 8 ցանցում գտնվող այլ հասցեները: Նկատենք, որ չնայած client ծրագրից bind() ֆունկցիան չկանչելուն, այն ավտոմատ կերպով կապվել է ժամանակավոր պորտի հետ (ephemeral port):

Ծրագիրը կատարելու համար անհրաժեշտ քայլերն են.

- Ստեղծել ծրագրերի կատարվող ֆայլերը.
  gcc ipv4_dg_server.c -o ipv4_dg_server
  gcc ipv4_dg_client.c -o ipv4_dg_client
- Կատարել ծրագրերն առանձին հրամանային տողերում.
  ./ipv4_dg_server
  ./ipv4_dg_client 127.0.0.1 hello

IPv6 սոկետի հասցեն

IPv6 սոկետի հասցեն պահվում է **sockaddr_in6** ստրուկտուրայում:

```
#include <netinet/in.h>
struct in6_addr { /* IPv6 address structure */
        uint8_t s6_addr[16]; /* 16 bytes == 128 bits */
};

struct sockaddr_in6 { /* IPv6 socket address */
        sa_family_t sin6_family; /* Address family (AF_INET6) */
        in_port_t sin6_port; /* Port number */
        uint32_t sin6_flowinfo; /* IPv6 flow information */
        struct in6_addr sin6_addr; /* IPv6 address */
        uint32_t sin6_scope_id; /* Scope ID (new in kernel 2.4) */
};
```

Ի տարբերություն IPv4-ի, որը 32 բիթ է, IPv6-ը ունի 128 բիթ երկարություն: **sin6_port** և **sin6_addr** դաշտերը պահում են պորտի համարը և IP հասցեն: Բոլոր դաշտերը ներկայացվում են ցանցի բայթերի կարգով: IPv6 wildcard հասցեի համար գրադարանում հայտարարված է **in6addr_any** փոփոխականը:

Ipv6_dg_server, ipv6_dg_client ծրագրերը

Այս ծրագրերը ներկայացնում են վերը դիտարկված խնդրի լուծումը IPv6 դոմեյնում: Client ծրագիրը որպես հրամանային տողի արգումենտ ընդունում է սերվերի IP հասցեն: Տվյալ օրինակում կիրառվում է ::1 հասցեն, որը համարվում է Ipv6 ցանցի հետադարձ հասցե (loopback address):

Ծրագիրը կատարելու համար անհրաժեշտ քայլերն են.

- Ստեղծել ծրագրերի կատարվող ֆայլերը.
  gcc ipv6_dg_server.c -o ipv6_dg_server
  gcc ipv6_dg_client.c -o ipv6_dg_client
- Կատարել ծրագրերն առանձին հրամանային տողերում.
  ./ipv6_dg_server
  ./ipv6_dg_client ::1 hello

Առաջադրանքներ

1. Կատարել ipv4_dg_server, ipv4_dg_client ծրագրերը:
2. Կատարել ipv6_dg_server, ipv6_dg_client ծրագրերը:
3. Հոսքային սոկետների միջոցով իրականացնել IPv4 ցանցում աշխատող Echo service: Server ծրագիրը պետք է միացումներ ընդունի տարբեր client-ներից և հետ ուղարկի ստացված տվյալները:
4. Հոսքային սոկետների միջոցով իրականացնել IPv4 ցանցում աշխատող chatroom: Server ծրագիրը պետք է միացումներ ընդունի տարբեր client-ներից և դրանցից յուրաքանչյուրի կողմից ստացված հաղորդագրությունը ուղարկի մյուս client-ներին: