Beej Guide To Network Programming

Beej Guide To Network Programming

General

- `AF_INET` refers to internet ipv4. `address family`. `PF_INET` and `protocol family` refers to anything with prococol or sockes/ports.
-

Sockets

- A way to speak to other programs using standard unix file descriptors
- File Descriptors are integer that is associated with open file. Different from PID(Process Identifier)
- Use `open()` or `socket()` and it will return a file descriptor which is a handle that is used to identify the file and the mode of operation.
- Mainly Two types of internet sockets
  - `SOCK_STREAM` : stream sockets
    - Reliable socket stream
    - connection must be made between sender and receiver.
    - Used for HTTP, telnet
    - Uses TCP
  - `SOCK_DGRAM` : datagram sockets
    - connectionless
    - used for `multiplayer games` ,
    - Uses UDP(User Datagram Protocol)
    - `ACK`(acknowledge signal) for reliable trasnportation

Difference between TCP/IP and TCP protocol

- TCP/IP is a stack with different layers where TCP is just one of its protocol in network transport layer.

Network Theory

- OSI model
  - `Please do not throw my salami pizza away`
  - Physical , data link layer, Network, transport, session , presentaion , application

- For UNIX network programming the simple model is

    - Application Layers
    - Host to host transport layer (TCP, UDP)
    - Internet Layer (IP and routing)
    - Netowk Access Layer(Ethernet , wifi etc)

IP Addresses , Structs And Data Mungling

- IPV4 vs IPv6
- 2^32 vs 2^128

Subnets

- seperation of `network portion` and `host portion`.
- Class A , B , C
    - Defined by netmask. If your netmask is `255.255.0.0`
    - Another way is `192.168.0.12/30` where 30 is network portion

Port numbers

- Besides IP there is another adress used by TCP and UDP. It is port number. 16 bit number
- hotel : IP address, room number: port
- `/etc/services` has list of all possible port numbers

Byte order

- Big Endian And Little Endian.
- Big Endian
    - if you want to store `b34f` first store b3 then store 4f . 0x2000 ma b3, 0x2001 ma 4f
    - `Network` byte order
- Little Endian
    - intel
    - `Host` Byte order

`htons()` `host` to `network` `short`

`htonl()` `host` to `network` `long`

`ntohl()` `network` to `host` `short`

Structs

- Socket Descriptor is `int`
- `struct addrinfo` : is more recent invention and is used to prep the socket address structures for subsequent use.

```
struct addrinfo {
    int ai_flags; // ai_passive, ai_cannonname
    int ai_family; // af_inet, af_inet6, af_unspec, af_bluetooth
    int ai_socktype; //sock_stream, sock_dgram
    int ai_protocol; //  use 0 for any
    size_t sockaddr *ai_addr; // struct sockaddr_in and _in6
    char *ai_canonname; //full canonical hostname

    struct addrinfo *ai_next; //linked list
}

struct sockaddr {
   unsigned short sa_family; // address family, AF_INET, AF_inet6
   char sa_data[14]; // 14 bytes of protocol address
}

// sockaddr and sockaddr_in are castable

struct sockaddr_in {
   short int sin_family; //address family, AF_INET
   unsigned short int sin_port; //port number
   struct in_addr sin_addr;
   unsigned char sin_zero[8];
}

struct in_addr {
   uint32_t s_addr; // 32-bit int(4 bytes)
}

//stores socket address information. can hold both ipv4 and ipv6
struct sockaddr_storage {
   short ss_family; //address family such as AF_INET.

   char __ss_pad1[_SS_PAD1SIZE]; // 48 bit pad that ensures 64 bit alignment
for ipv4. 16 remain which is aligned.
   int64_t __ss_align;
   char __ss_pad2[_SS_PAD2SIZE];
```

```
}
```

IP Addresses

```
struct sockaddr_in sa;
struct sockaddr_in6 sa6;

//convert IPV4 and IPV6 from text to binary form.
//presentation to network
inet_pton(AF_INET, "10.12.110.57", &(sa.sin_addr));


//network to presentation
//ntop
inet_ntop(AF_INET6, &(sa6.sin6_addr), ip6, INET6_ADDRSTRLEN);
printf("%s\n",ip6);
```

private (or disconnected) networks

- starts with `10.0` or `192.`
- for ipv6 starts with `fdXX` where X is hexadecimal notation.

System calls or Bust

- `getaddrinfo()` - prepare to launch.