Guide to Socket Programming

**Types of sockets**

1) Stream Socket (SOCK\_STREAM) – Connected socket - Reliable

2) Datagram Socket (SOCK\_DGRAM) – Connection-less socket – Unreliable

How to get a socket ?

You make a call to the socket()system routine. It returns the socket descriptor, and you communicate through it using the specializedsend() and recv() (man send, man recv) socket calls.

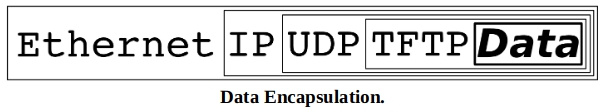
**Stream Sockets**

* Uses TCP – Transmission Control Protocol
* TCP makes sure your data arrives sequentially and error-free
* Uses IP for routing
* IP deals primarily withInternet routing and is not generally responsible for data integrity

Examples of stream sockets – Telnet; HTTP protocol uses stream sockets

**Datagram Sockets**

* Uses UDP - “User DatagramProtocol”
* you don't have to maintain an open connection as you do with stream sockets. You just build a packet, slap an IP header on it with destination information, and send it out. No connection needed.
* Uses IP for routing
* Does not use TCP
* Datagram sockets have acknowledgment procedure (implemented on top of UDP) to ensure all data is received properly.
* Higher speed compared to TCP



**A layered model more consistent with Unix might be:**

**Application Layer (telnet, ftp, etc.)**

**Host-to-Host Transport Layer (TCP, UDP)**

**Internet Layer (IP and routing)**

**Network Access Layer (Ethernet, wi-fi, or whatever)**

**IP address**

**IPv4** (192.0.2.111)

32 bits

Approx 4 billion address

4 octets

127.0.0.1 – Loop-back address

**IPv6** (2001:0db8:c9d2:0012:0000:0000:0000:0051) (2001:db8:c9d2:12::51)

128 bits

::1 – Loop-back address

IP address has - [Network portion : Host Portion]

**Port Numbers**

* Well known port range 0 – 1023
* Registered ports are 1024 to 49151
* Dynamic ports 49152 to 65535

80 – HTTP

443 – HTTPS

22 – SSH

20 – FTP—Data

23 – Telnet

<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

**Byte order**

Big-Endian – Network Byte order

Little-Endian – Host Byte order - Most systems

short - 2 bytes (16 bit)

long - 4 bytes (32 bit)

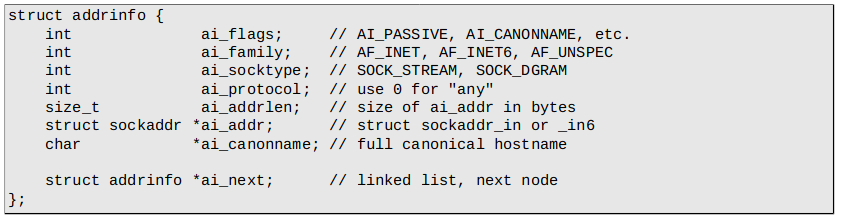
htons() - host tonetwork short

htonl() - host tonetwork long

ntohs() - network tohost short

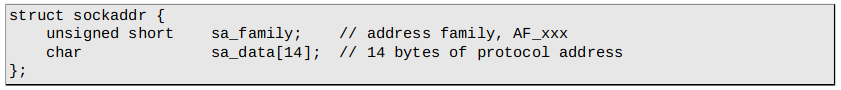
ntohl() - network tohost long

**structs**



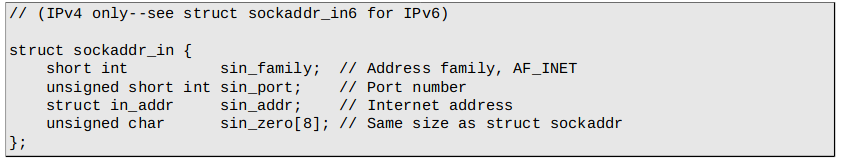
ai\_family = IPv4 (AF\_INET) or IPv6 (AF\_INET6) or whatever

ai\_next = returned linked list from getaddrinfo()



sa\_family = IPv4 or IPv6 or whatever

sa\_data = destination address and port number

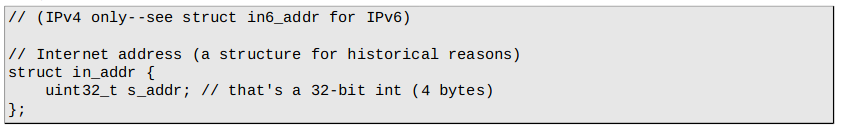


**parallel structure to sockaddr;** To avoid tedious packing of sa\_data in sockaddr.

sin\_family = IPv4 or IPv6 or whatever

sin\_zero should be set to all zero using memset() function

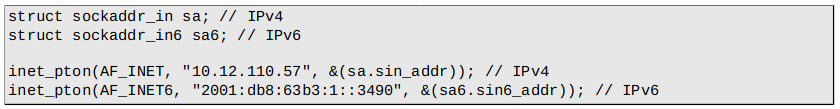
sin\_port should be in network byte order



sin\_add ‘s type

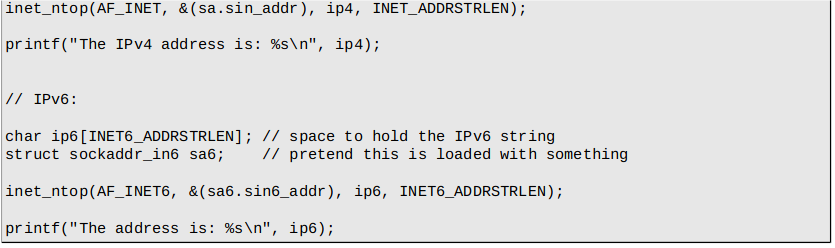
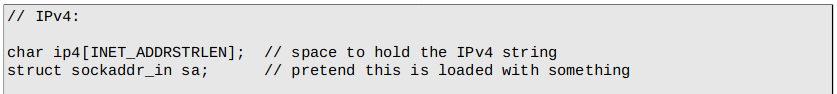
s\_addr should be in network byte order

**IP address manipulation**



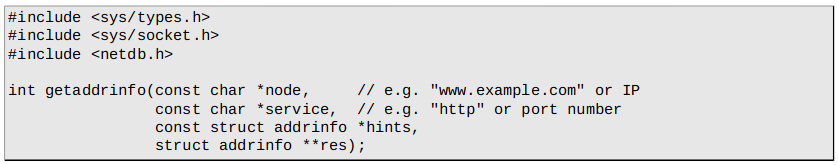
No error checking in inet\_pton (-1 error; 0 messed up IP)

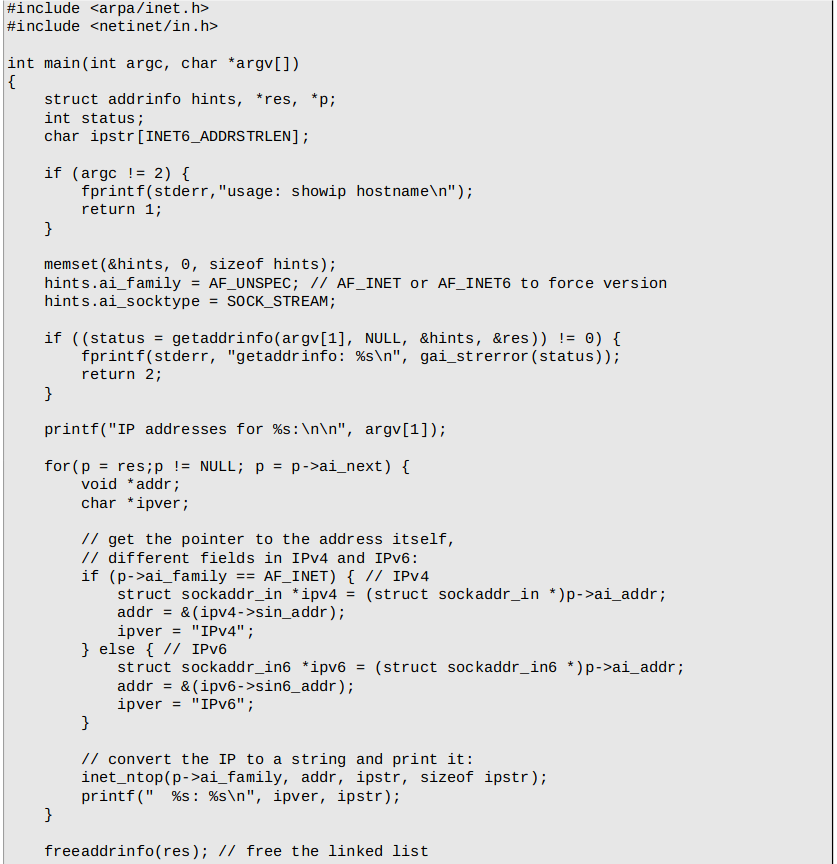
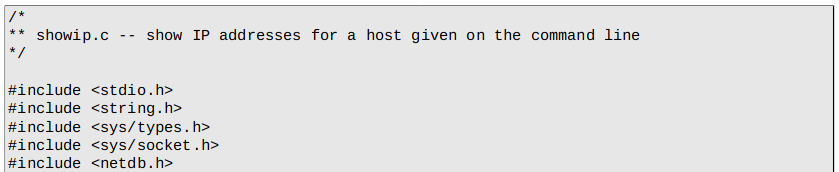
pton – Presentation to network



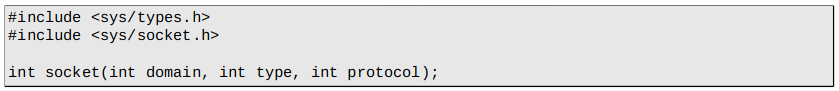
only works with numeric IP

**getaddrinfo()**





**socket()**



domain – PF\_INET, PF\_INET6

type – SOCK\_STREAM, SOCK\_DGRAM

protocol – 0 for any (same as ai\_protocol in addrinfo)or getprotobyname()

PF\_INET is similar to AF\_INET

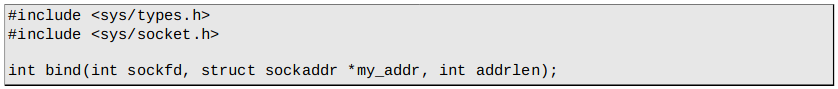
AF stands for Address Family

PF stands for Protocol Family

returns -1 on errors and sets errno to error’s value

sockfd = socket(res->ai\_family, res->ai\_socktype, res->ai\_protocol);

**bind()**



Associates socket to a port

Used only by the server (No need for a client program)

sockfd – socket file descriptor returned by socket()

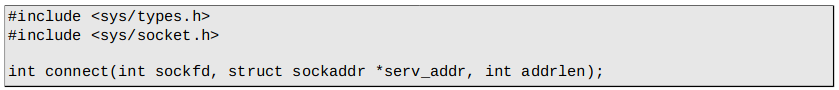
my\_addr – contains information about your address – port and IP address

addrlen – length in bytes of that address

returns -1 on errors and sets errno to error’s value

bind(sockfd, res->ai\_addr, res→ai\_addrlen)

**connect()**



sockfd – socket file descriptor returned by socket()

serv\_addr – destination port and ip address

addrlen – Length of bytes of serv\_addr

connect(sockfd, res->ai\_addr, res→ai\_addrlen);

**listen()**

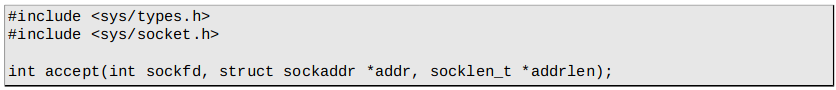


sockfd – socket file descriptor returned by socket()

backlog – Number of connections allowed on incoming queue

returns -1 on errors and sets errno to error’s value

**accept()**

****

Returns a new socket file descriptor.

returns -1 on errors and sets errno to error’s value