Introduction to Network Programming - continuation

Lecture 2

Byte manipulation functions

- inet_aton()
- inet_addr()
- inet_ntoa()

- inet_pton()
- inet_ntop()

Header file is ***<arpa/inet.h>***

- For the functions : inet_aton(), inet_addr(), inet_ntoa()

- Convert IPv4 address between a dotted-decimal string

(eg. “206.90.220.10”) and its 32-bit network byte ordered binary value

inet_aton()

- Convert from a dotted decimal string (d.d.d.d) pointed to by *strptr* into a 32-bit binary network byte ordered value which is stored through the pointer *addrptr*

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

- Returns 1 if string is valid else 0 on error

inet_ntoa()

- Convert a 32-bit binary network byte ordered into its corresponding dotted-decimal string

```
char *inet_ntoa(struct in_addr inaddr);
```

- Returns pointer to dotted-decimal string

inet_addr()

- Not much usage of this function
- Returns 32-bit binary network byte ordered IPv4 address; INADDR_NONE if error

```
in_addr_t inet_addr(const char *strptr);
```

For reference..


```
struct in_addr
{
    in_addr_t s_addr; /* 32-bit IP address */
};
```

```
struct sockaddr_in
{
    uint8_t sin_len; //length of structure
    sa_family_t sin_family; // AF_INET
    in_port_t sin_port; //TCP or UDP port
    struct in_addr sin_addr; //IP address
    char sin_zero[8]; //unused
};
```

Example Program


```
struct sockaddr_in a;
char *some_addr;
inet_aton("10.0.0.1", &a.sin_addr);
// store IP in sin_addr
some_addr =inet_ntoa(a.sin_addr);
// return the IP
printf("%s\n", some_addr); // prints "10.0.0.1"

//same as the inet_aton() call, above:
a.sin_addr.s_addr = inet_addr("10.0.0.1");
```

Check if IPv4 address is valid


```
#include "unp.h"
int main(int argc, char **argv)
{
        struct in_addr addr;
        if (argc!=2){
                printf("%s <dotted-address>\n", argv[0]);
                exit(1); //Exit failure
        }
        if(inet_aton(argv[1], &addr)==0){
                printf("Address invalid...\n\n");
                exit(1);
        }
        printf("\n The address is valid: \t");
        printf("%s \n\n\n", inet_ntoa(addr));
        exit(0); //Exit success
}
```

Read and convert the IPv4 address


```
#include "unp.h"

int main(int argc, char *argv[])
{
        struct in_addr addr;
        char *a;
        inet_aton(argv[1], &addr); //store IP address in addr
        a=inet_ntoa(addr);  //return IP
        printf("After conversion = %s\n\n", a);
        exit(0);
}
```

inet_pton() and inet_ntop()

- New with IPv6 and work with both IPv4 and IPv6 addresses
- Letters p stands for presentation and n for numeric
- Presentation format is binary value that goes into a socket address structure

inet_ntop()

- Converts from numeric to presentation

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

- *family* - address family parameter (either AF_INET or AF_INET6)
- *addrptr* - pointer to either a struct in_addr or struct in6_addr containing the address to convert to a string
- *strptr* - pointer to the destination string. It cannot be a null pointer
- *len* - maximum length of that string.

- Caller must allocate memory for destination and specify its size
- It returns the *addrptr* parameter on success, or NULL on failure

inet_pton()

- Convert a string pointed to by *strptr*, storing binary result through pointer *addrptr*

int inet_pton(int family, const char *strptr, void *addrptr);

- *family* - address family (either AF_INET or AF_INET6).
- *strptr* - pointer to a string containing the IP address in printable form.
- *addrptr* - points to where the result should be stored, which is probably a struct in_addr or struct in6_addr
- Returns 1 on success (network address was successfully converted). It returns -1 on error, or 0 if the input isn't a valid IP address.

Example Program


```
// IPv4 demo of inet_ntop() and inet_pton()
struct sockaddr_in sa;
char str[INET_ADDRSTRLEN]; //16

// store this IP address in sa:
inet_pton(AF_INET, "192.0.2.33", &(sa.sin_addr));

// now get it back and print it
inet_ntop(AF_INET, &(sa.sin_addr), str, INET_ADDRSTRLEN);

printf("%s\n", str);
// prints "192.0.2.33"
```

Example Program


```
// IPv6 demo of inet_ntop() and inet_pton()
struct sockaddr_in6 sa;
char str[INET6_ADDRSTRLEN]; //46

// store this IP address in sa:
inet_pton(AF_INET6, "2001:db8:8714:3a90::12", &(sa.sin6_addr));

// now get it back and print it
inet_ntop(AF_INET6, &(sa.sin6_addr), str, INET6_ADDRSTRLEN);

printf("%s\n", str);
// prints "2001:db8:8714:3a90::12"
```

Check if IPv4 address is valid


```
#include "unp.h"
#define INET_ADDRSTRLEN 16 //for IPv4 dotted-decimal
#define INET6_ADDRSTRLEN 46       //for IPv6 hex string

int main(int argc, char *argv[])
{
        struct sockaddr_in addr;
        char str[INET_ADDRSTRLEN];
        char *ptr;
        if (argc!=2)
        {
                printf("%s <IP address> \n\n", argv[0]);
                exit(1);
        }
```

Check if IPv4 address is valid …


```
        if (inet_pton(AF_INET, argv[1], &addr.sin_addr)!=1)
        {
                printf("Error : Not valid presentaion");
                exit(1);
        }

        inet_ntop(AF_INET, &addr.sin_addr, str, sizeof(str));

        printf("IP address = %s\n\n", str);
        exit(0);
}
```

How would you convert from IPv4 to IPv6 address?

- Ask from user the valid IPv4 address (hint: inet_pton)
- Convert it to IPv6 address (hint: inet_ntop)