Prev Contents Next

struct sockaddr_in, struct in_addr

Structures for handling internet addresses

Prototypes

```
#include <netinet/in.h>

struct sockaddr_in {
    short            sin_family;   // e.g. AF_INET
    unsigned short   sin_port;     // e.g. htons(3490)
    struct in_addr   sin_addr;     // see struct in_addr, below
    char             sin_zero[8];  // zero this if you want to
};

struct in_addr {
    unsigned long s_addr;  // load with inet_aton()
};
```

Description

These are the basic structures for all syscalls and functions that deal with internet addresses. In memory, the `struct sockaddr_in` is the same size as `struct sockaddr`, and you can freely cast the pointer of one type to the other without any harm, except the possible end of the universe.

Just kidding on that end-of-the-universe thing...if the universe does end when you cast a `struct sockaddr_in*` to a `struct sockaddr*`, I promise you it's pure coincidence and you shouldn't even worry about it.

So, with that in mind, remember that whenever a function says it takes a `struct sockaddr*` you can cast your `struct sockaddr_in*` to that type with ease and safety.

There's also this `sin_zero` field which some people claim must be set to zero. Other people don't claim anything about it (the Linux documentation doesn't even mention it at all), and setting it to zero doesn't seem to be actually necessary. So, if you feel like it, set it to zero using `memset()`.

Now, that `struct in_addr` is a weird beast on different systems. Sometimes it's a crazy `union` with all kinds of `#define`s and other nonsense. But what you should do is only use the **`s_addr`** field in this structure, because many systems only implement that one.

With IPv4 (what basically everyone in 2005 still uses), the `struct s_addr` is a 4-byte number that represents one digit in an IP address per byte. (You won't ever see an IP address with a number in it greater than 255.)

Example

```
struct sockaddr_in myaddr;
int s;

myaddr.sin_family = AF_INET;
myaddr.sin_port = htons(3490);
inet_aton("63.161.169.137", &myaddr.sin_addr.s_addr);

s = socket(PF_INET, SOCK_STREAM, 0);
bind(s, (struct sockaddr*)myaddr, sizeof(myaddr));
```

See Also

accept(), bind(), connect(), inet_aton(), inet_ntoa()

Prev Contents Next