

# Networking, Part 5: Reusing ports

SurtaiHan edited this page 7 days ago · 6 revisions

## When I re-run my server code it doesn't work! Why?

By default, after a socket is closed the port enters a time-out state during which time it cannot be re-used ('bound to a new socket').

This behavior can be disabled by setting the socket option REUSEPORT before bind-ing to a port:

```
int optval = 1;
setsockopt(sock_fd, SOL_SOCKET, SO_REUSEPORT, &optval, sizeof(optval));

bind(sock_fd, ...);
```

## Can a TCP client bind to a particular port?

Yes! In fact outgoing TCP connections are automatically bound to an unused port on the client. Usually it's unnecessary to explicitly set the port on the client because the system will intelligently find an unused port on a reasonable interface (e.g. the wireless card, if currently connected by WiFi connection). However it can be useful if you needed to specifically choose a particular ethernet card, or if a firewall only allows outgoing connections from a particular range of port values.

To explicitly bind to an ethernet interface and port, call `bind` before `connect`

## Who connected to my server?

The `accept` system call can optionally provide information about the remote client, by passing in a `sockaddr` struct. Different protocols have differently variants of the `struct sockaddr`, which are different sizes. The simplest struct to use is the `sockaddr_storage` which is sufficiently large to represent all possible types of `sockaddr`. Notice that C does not have any model of inheritance. Therefore we need to explicitly cast our struct to the 'base type' `struct sockaddr`.

```
struct sockaddr_storage clientaddr;
socklen_t clientaddrsz = sizeof(clientaddr);
int client_id = accept(passive_socket, (struct sockaddr *) &clientaddr, &clie
```

We've already seen `getaddrinfo` that can build a linked list of `addrinfo` entries (and each one of these can include socket configuration data). What if we wanted to turn socket

Edit

New Page

▼ Pages 51

[Home](#)

[#Example Markdown](#)

[#Informal Glossary](#)

[#Piazza: When And How to Ask For Help](#)

[C Programming, Part 1: Introduction](#)

[C Programming, Part 2: Text Input And Output](#)

[C Programming, Part 3: Common Gotchas](#)

[C Programming, Part 4: Debugging](#)

[Deadlock, Part 1: Resource Allocation Graph](#)

[Deadlock, Part 2: Deadlock Conditions](#)

[File System, Part 1: Introduction](#)

[File System, Part 2: Files are inodes \(everything else is just data...\)](#)


[File System, Part 3: Permissions](#)

[File System, Part 4: Working with directories](#)

[File System, Part 5: Virtual file systems](#)

[Show 36 more pages...](#)

Clone this wiki locally

 Clone in Desktop

data into IP and port addresses? Enter `getnameinfo` that can be used to convert a local or remote socket information into a domain name or numeric IP. Similarly the port number can be represented as a service name (e.g. "http" for port 80). In the example below we request numeric versions for the client IP address and client port number.

```
socklen_t clientaddrsz = sizeof(clientaddr);
int client_id = accept(sock_id, (struct sockaddr *) &clientaddr, &clientaddrsz);
char host[256], port[256];
getnameinfo((struct sockaddr *) &clientaddr, clientaddrsz, host, sizeof(host),
```

Todo: Discuss NI\_MAXHOST and NI\_MAXSERV, and NI\_NUMERICHOST

## getnameinfo Example: What's my IP address?

To obtain a linked list of IP addresses of the current machine use `getifaddrs` which will return a linked list of IPv4 and IPv6 IP addresses (and potentially other interfaces too). We can examine each entry and use `getnameinfo` to print the host's IP address. The `ifaddrs` struct includes the family but does not include the `sizeof` of the struct. Therefore we need to manually determine the struct sized based on the family (IPv4 v IPv6)

```
(family == AF_INET) ? sizeof(struct sockaddr_in) : sizeof(struct sockaddr_in6)
```

The complete code is shown below.

```
int required_family = AF_INET; // Change to AF_INET6 for IPv6
struct ifaddrs *myaddrs, *ifa;
getifaddrs(&myaddrs);
char host[256], port[256];
for (ifa = myaddrs; ifa != NULL; ifa = ifa->ifa_next) {
    int family = ifa->ifa_addr->sa_family;
    if (family == required_family && ifa->ifa_addr) {
        if (0 == getnameinfo(ifa->ifa_addr,
                               (family == AF_INET) ? sizeof(struct sockaddr_in)
                               : sizeof(struct sockaddr_in6),
                               host, sizeof(host), port, sizeof(port), NI_NUMERICHOST, 0))
            puts(host);
    }
}
```

## What's my machine's IP address (shell version)

Answer: use `ifconfig` (or Windows's `ipconfig`) However this command generates a lot of output for each interface, so we can filter the output using `grep`

```
ifconfig | grep inet

Example output:
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
inet 127.0.0.1 netmask 0xff000000
inet6 ::1 prefixlen 128
inet6 fe80::7256:81ff:fe9a:9141%en1 prefixlen 64 scopeid 0x5
inet 192.168.1.100 netmask 0xfffff00 broadcast 192.168.1.255
```

Legal and Licensing information: Unless otherwise specified, submitted content to the wiki must be original work (including text, java code, and media) and you provide this material under a [Creative Commons License](#). If you are not the copyright holder, please give proper attribution and credit to existing content and ensure that you have license to include the materials.

