1. Review: What is htons? ntohs? Why do we need them? What do their names stand for?

What are the "four calls" to set up the server? What is their order? And what is their purpose?

Quick comment: How to use freeaddrinfo struct addrinfo hints, *result;
memset etc
getaddrinfo(addr_string, port_string, &hints, &result);
freeaddrinfo(result);

2. What is port hijacking? What steps does the O/S take to prevent port hijacking?

Writing high-performance servers; handling 1000s of concurrent sockets The *select – poll – epoll* story

Differences between select and epoll? When would you use select?

3. Useful Socket/Port Know-how for developers

1) When I restart my program how can I reuse the same port immediately?

2) Creating a server that runs on an arbitrary port?
getaddrinfo(NULL, **"0",** &hints, &result); // ANY Port
Later...
struct sockaddr_in sin;
socklen_t socklen = sizeof(sin);
if (getsockname(sock_fd, (struct sockaddr *)&sin, &socklen) == 0) printf("port %d\n", sin.sin_port);
// Hint: Something is missing above here

4. Client IP address?

```
struct sockaddr_in client_info;
int size = sizeof(client_info);
int client_fd = accept(sock_fd, (struct sockaddr*) &client_info, &size);

 char *connected_ip= inet_ntoa(client_info.sin_addr);  // Does this look thread-safe to you?
 int port = ntohs(client_info.sin_port);
 printf("Client %s port %d\n", connected_ip, port);
```

5. Build a non-compliant web server!

Send some text ....

```
read(client_fd, buffer, ...);

dprintf(client_fd,"HTTP/1.0 200 OK\r\n"
      "Content-Type: text/html\r\n"
      "Connection: close\r\n\r\n");

dprintf(client_fd,"<html><body><h1>Hello!");
dprintf(client_fd,"</h1></body></html>");


shutdown(client_fd , SHUT_RDWR)
close(client_fd);
```

Send a picture

```
read(client_fd, buffer, ...);
```

Epoll notes