CS341 #33- Epoll. Server & Port know-how. Web server.

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?

server : socket bind listen accept shutdown() & close()

client : socket connect

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

select is portable

epoll for the win! (Linux)

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

epoll : (1) edge triggered only new stuff

(2) level triggered warning all the time

↑

use this!

3. Useful Socket/Port Know-how for developers

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

setsockoption(fd, SOL_SOCKET, SO_REUSEADDR, &opt

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

ntohs

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

select:
old, cross-platform - Even available on tiny embedded linux systems
Requires simple but O(N) linear scan- so does not scale well
Hard-limit on number of selectors
<1ms timeout

poll
Also O(N) scan
OSX support
Good for short-lived sockets or 100s of sockets
can detect closed sockets
1ms+ timeout
Cannot close sockets during poll
event based

epoll – newest. linux specific; not Macosx (use kqueue instead)
good for large (1000s) of long-lived sockets per thread
long-lived = multi I/O requests per connection
1ms+ timeout
event based
Each accept'ed call needs to be added to the set

.. what if I have 100s of long-lived sockets on Linux? poll vs epoll? Ans: There may not be a significant difference between either approach. Try both and benchmark.

An excellent in-depth article about the differences between select, poll and epoll:

http://www.ulduzsoft.com/2014/01/select-poll-epoll-practical-difference-for-system-architects/

stat
```
char*buf = malloc(st.size);
fread(buf,1,st.size,file);

dprintf(client_fd,"HTTP/1.1 200 OK\r\nContent-Type: image/jpeg\r\n");
dprintf(client_fd,"Content-Length: %ld\r\n\r\n",size);

write(client_fd, buf, size);
fclose(file);
free(buf);
```