

Fun with `python3 -m http.server (SimpleHTTPServer)`, `telnet towel.blinkenlights.nl` and `netcat`

1. TCP Client (Review) + IPv6

1.1 What are the steps to setting up a client TCP socket?

i. _____ ii. _____ iii. _____

1.2 How many `addrinfo` structs does `getaddrinfo` return? Why?

1.3 How do I get a string error with `getaddrinfo` returns?

1.4 What is `AF_INET6`?

1.5 What is `0:0:0:0:0:0:1`?

1.6 Using `getaddrinfo` how do I ask for stream-based https IP4?

```
int startserver() {
    struct addrinfo hints, *result;
```

```
    _____

    hints.ai_family    = _____
```

```
    hints.ai_socktype = _____
```

```
    int result = _____(_____, _____, _____, _____) ?
```

```
}
```

1.7 For each `addrinfo` what do you call next?

1.8 Can you `bind()` a client socket? Why would you want to?

```
struct addrinfo {
    int          ai_flags
    int          ai_family
    int          ai_socktype
    int          ai_protocol
    socklen_t    ai_addrlen
    struct sockaddr *ai_addr
    char         *ai_canonname
    struct addrinfo *ai_next
}
```

2. TCP SERVER

2.1 What is a passive socket? How do you specify it?

2.2 Why would I create one?

2.3 If you don't bind what do you get?

2.4 What is `htons`? `ntohs`? Why/when do we need them?

```
struct sockaddr_in stSockAddr;
int SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
memset(&stSockAddr, 0, sizeof(stSockAddr));
stSockAddr.sin_family = AF_INET;
stSockAddr.sin_port = htons(1100);
stSockAddr.sin_addr.s_addr = htonl(INADDR_ANY);
```

2.5 **Important!** What are the "*four calls*"? What is their order? And what is their purpose?

i. _____ ii. _____ iii. _____ iv. _____

```

#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>
//plus string.h, stdlib.h stdio.h

int main(int argc, char** argv) { // TCP Server
    int s;
    int sock_fd = socket(AF_INET, SOCK_STREAM, 0);

    struct addrinfo hints, *result;
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;

    s = getaddrinfo(NULL, "1234", &hints, &result);
    if (s != 0) {
        fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
        exit(1);
    }

    if ( bind(sock_fd, result->ai_addr, result->ai_addrlen) != 0 ) {
        perror("bind()"); exit(1);
    }
    if ( listen(sock_fd, 10) != 0 ) {
        perror("listen()"); exit(1);
    }

    struct sockaddr_in * result_addr = (struct sockaddr_in*) result->ai_addr;
    printf("Listening on file descriptor %d, port %d\n", sock_fd, ntohs(result_addr->sin_port));

    printf("Waiting for connection...\n");
    int client_fd = accept(sock_fd, NULL, NULL);
    printf("Connection made: client_fd=%d\n", client_fd);

    char buffer[1000];
    int len = read(client_fd, buffer, 999);
    printf("Read %d chars\n", len);
    if( len > 0 ) {
        buffer[len] = '\0';
        printf("%s\n", buffer);
    }
    return 0;
}

```

Limitations: No SIGPIPE support. Single threaded. No port reuse.
 If there's time...
 What is a 'honey pot? What is `epoll`? What is `select`?