Introduction to network programming in C

- Useful for the first project
- IPv6 and UDP

Basic functions

- `getaddrinfo()`
- `socket()`
- `bind()`
- `connect()`
- `send() / recv()`
- `select()`

Address resolution

- `getaddrinfo()`
- Input: domain name or textual IP address
- Output: IP address in binary form

Creating a socket

- `socket()`
- Input: Family (IPv4, IPv6, etc), type (stream (TCP), datagram (UDP), etc)
- Output: descriptor (socket)

Binding to a local address

- `bind()`
- Input: socket, address
- Output: error code, if no error then socket is bound
- Effect: transmitted packets have specified address as source

Connecting

- `connect()`
- Input: socket, address
- Output: error code
- Effect for UDP: transmitted packets through this socket have specified address as destination

Sending data

- `send()`
- Input: socket, data
- Output: error code
- Effect: data is transmitted to previously specified destination address

Receiving data

- `recv()`
- Input: socket
- Output: received data
- Effect: if a packet is available, then its data is written in the specified buffer

Non-blocking functions

- `recv()` will block until data is available
- To avoid this:
  - `fcntl(sock, F_SETFL, O_NONBLOCK);`
- `recv()`will return `EAGAIN` if no data is available

Multiplexing

- `select()`
- Input: descriptors, timeout
- Output: descriptors available for reading and/or writing, or nothing if call timed out
- Allows to do something else while periodically checking if something is available (typically used in a loop)

Representing data

- Network data is big endian
- Example, the 16-bit number 14344 is represented as `00111000 00001000`
- In Intel architecture, the number is encoded with the least significant byte first, then the most significant
- In network byte order, the most significant byte is first

Converting data

- `htons() / htonl()`
- `ntohs() / ntohl()`

Structures

-
  ```
  struct blah {
    uint8_t a : 6;
    uint8_t b : 2;
    uint16_t c;
  } __attribute__((packed));
  ```
- a and b are part of the same byte, but a uses the first 6 bits and b uses the last 2 bits
- The `packed` attribute says that the compiler must not insert padding for memory alignment

For more information

- RTFM
- STFW

Good luck !