CS 392 - Socks

- Dr. Shudongo
- Like fifo but for between machines
- The project!

Connection types

- Connection-oriented
    - Like calling, other end needs to pick up to work
    - TCP <-- we're doing this one
- Connection-less
    - basically a mailbox
    - UDP

Address

- 32-bit int (IPV4)
- Port number - :)
    - 65535 ports on default linux machine
    - Special ports used by system
    - 1-->1023: Used by system
    - 1024-->49151: user ports
    - 49152 --> 65535: Also system
    - Check with `cat /etc/services`

Domain / Protocol

- yippee
- (Connection types)
- Socket type and connection type must match

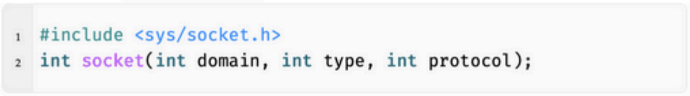
Stuff to know:

1. Can be used on same host
2. bidirectional

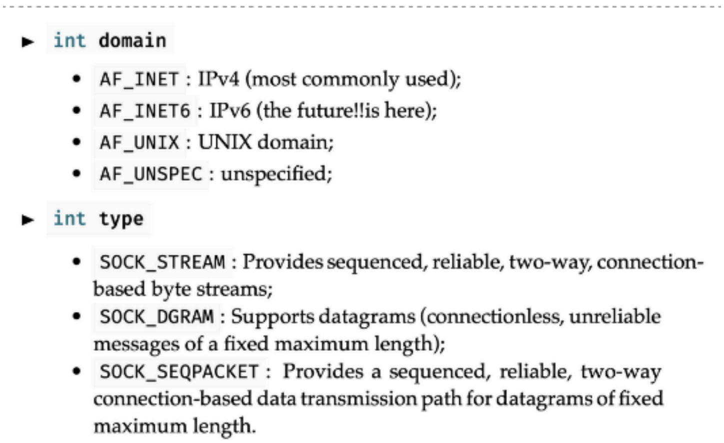
How to socket

Server side

- Create using `socket()`



```c
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```



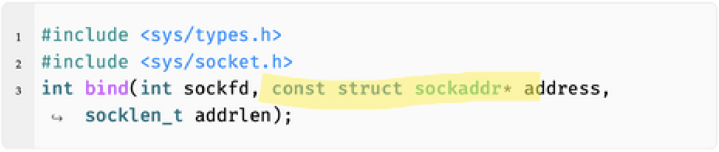
► `int domain`

- `AF_INET` : IPv4 (most commonly used);
- `AF_INET6` : IPv6 (the future!!is here);
- `AF_UNIX` : UNIX domain;
- `AF_UNSPEC` : unspecified;

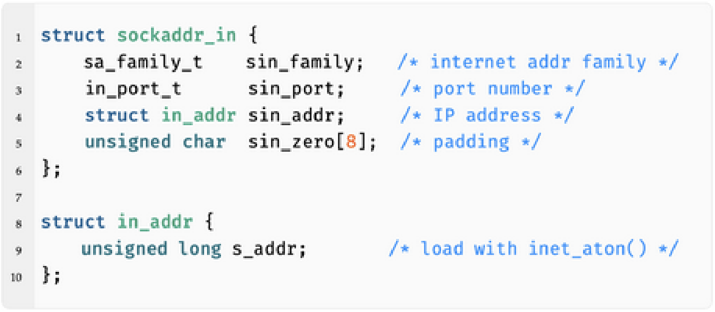
► `int type`

- `SOCK_STREAM` : Provides sequenced, reliable, two-way, connection-based byte streams;
- `SOCK_DGRAM` : Supports datagrams (connectionless, unreliable messages of a fixed maximum length);
- `SOCK_SEQPACKET` : Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length.

- Bind using `bind()`
  - Gives socket a "name"

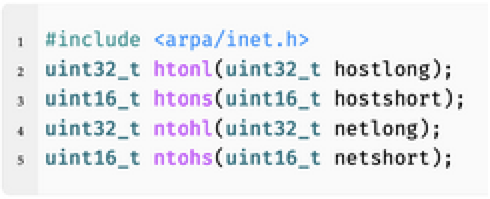
- 

```c
#include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd, const struct sockaddr* address,
    socklen_t addrlen);
```

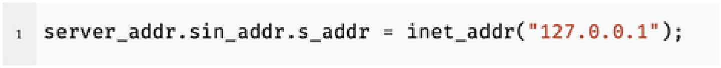
- 

```c
struct sockaddr_in {
    sa_family_t    sin_family;  /* internet addr family */
    in_port_t      sin_port;    /* port number */
    struct in_addr sin_addr;    /* IP address */
    unsigned char  sin_zero[8]; /* padding */
};

struct in_addr {
    unsigned long s_addr;      /* load with inet_aton() */
};
```

- 

```c
#include <arpa/inet.h>
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

- 

```c
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

- Listen `listen()` to monitor incoming stuff
- Loop where it accepts `accept()` new connections and does stuff with data

```c
// create socket
sock_fd = socket(AF_INET, ...); // unfinshed

memset(&addr, 0, sizeof(addr)) // zero out memory
addr.sin_fanily = AF_INET;
addr.sin_addr.s_addr = inet_addr(<ip>);
addr.sin_port = htons(<num>);

```

```c
bind(s, (struct socketaddr*) &addr),
sizeof(addr));
```

Client side