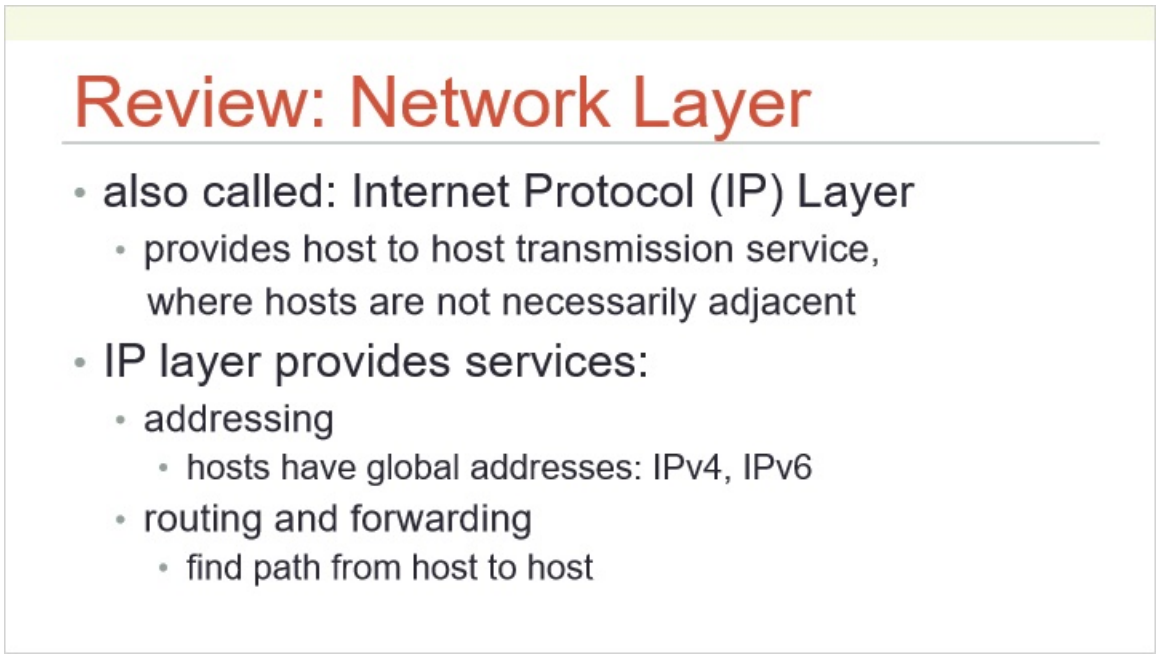
UDP

1. User Datagram Protocol

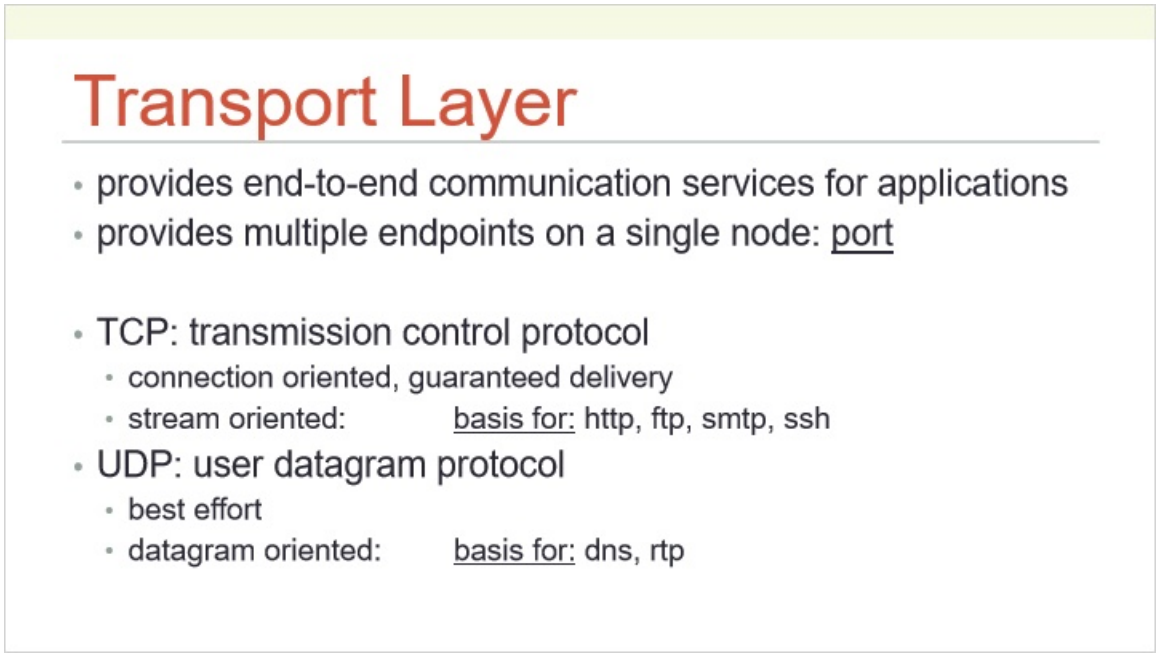
1.1 CSCI 330

1.2 Unit Overview

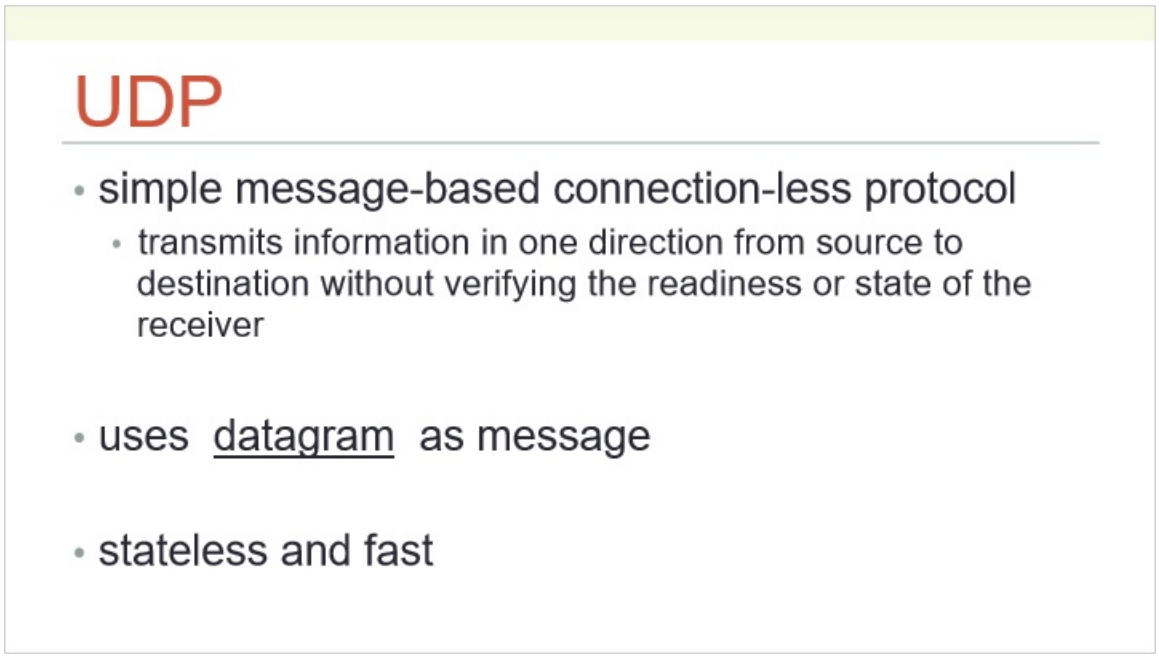
1.3 Review: Network Layer



1.4 Transport Layer



1.5 UDP



1.6 UDP packet format

UDP packet format

bits	0 – 7	8 – 15	16 – 23	24 – 31
0	Source IP address			
32	Destination IP address			
64	Zeros	Protocol	UDP length	
96	Source Port		Destination Port	
128	Length		Checksum	
160+	Data			

1.7 UDP programming

UDP programming

- common abstraction: socket
  - first introduced in BSD Unix in 1981
- socket is end-point of communication link
  - identified as IP address + port number
  - can receive data, can send data
- typical logic: server vs. client
  - server ready to receive datagram from any client
  - client sends datagram to specific server
  - server responds with datagram to client

1.8 Socket system calls

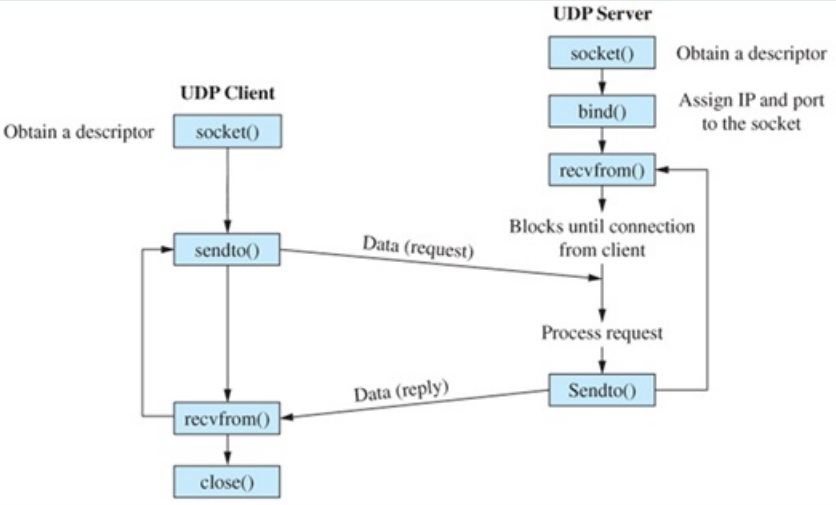
Socket system calls

server client

System call	Meaning
socket	Create a new communication endpoint
bind	Attach a local address to a socket
recvfrom	Receive(read) some data over the connection
sendto	Send(write) some data over the connection
close	Release the connection

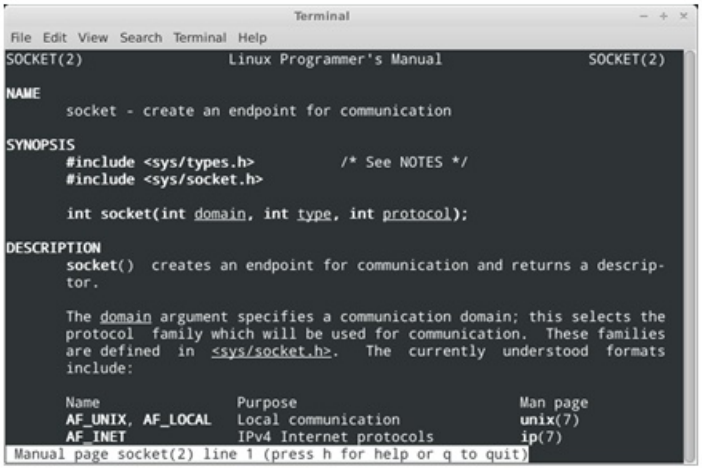
1.9 UDP communications pattern

UDP communications pattern



1.10 System call: socket

System call: socket



1.11 System call: socket

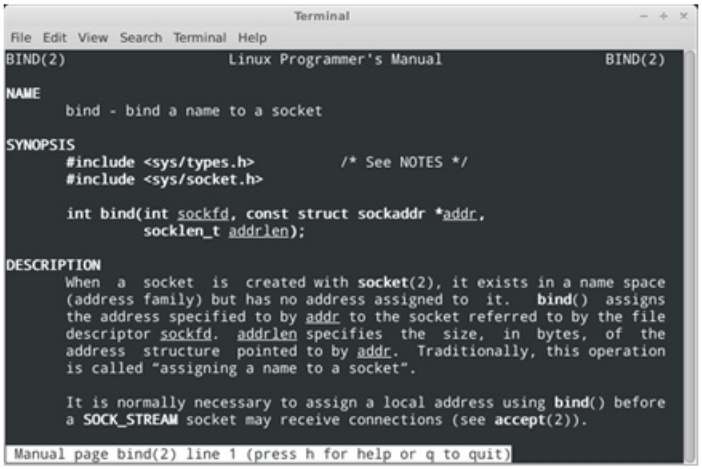
System call: socket

`int socket(int domain, int type, int protocol)`

- creates a new socket, as end point to a communications link
- **`domain`** is set to **`AF_INET`**
- **`type`** is set to **`SOCK_DGRAM`** for datagrams
- **`protocol`** is set to 0, i.e. default UDP
- returns socket descriptor:
  - used in **`bind`**, **`sendto`**, **`recvfrom`**, **`close`**

1.12 System call: bind

System call: bind



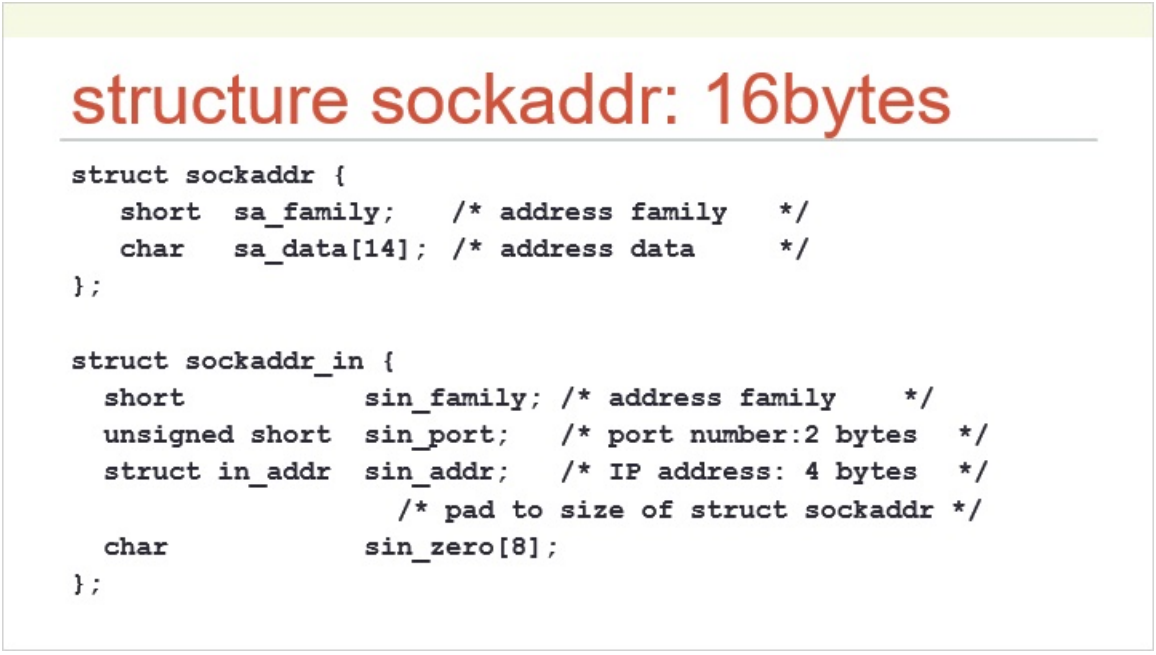
1.13 System call: bind

System call: bind

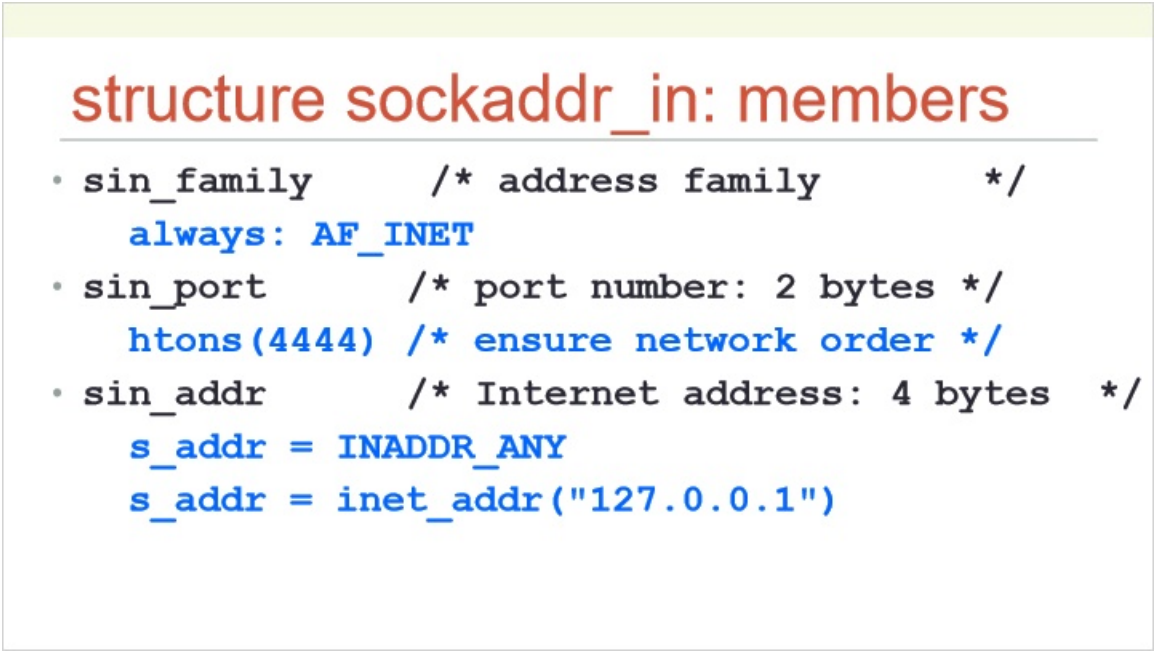
```
int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen)
```

- assigns address to socket: IP number and port
- **`struct sockaddr`** holds address information
  - will accept **`struct sockaddr_in`** pointer
- **`addrlen`** specifies length of **`addr`** structure
- returns 0 on success, -1 otherwise

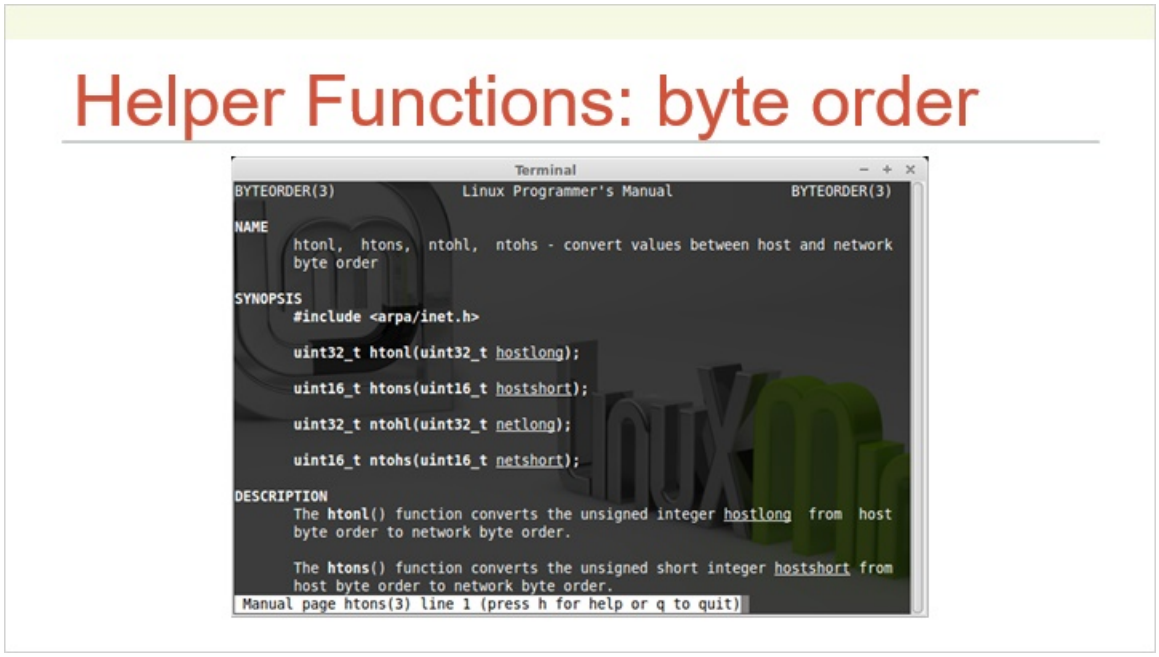
1.14 structure sockaddr: 16bytes



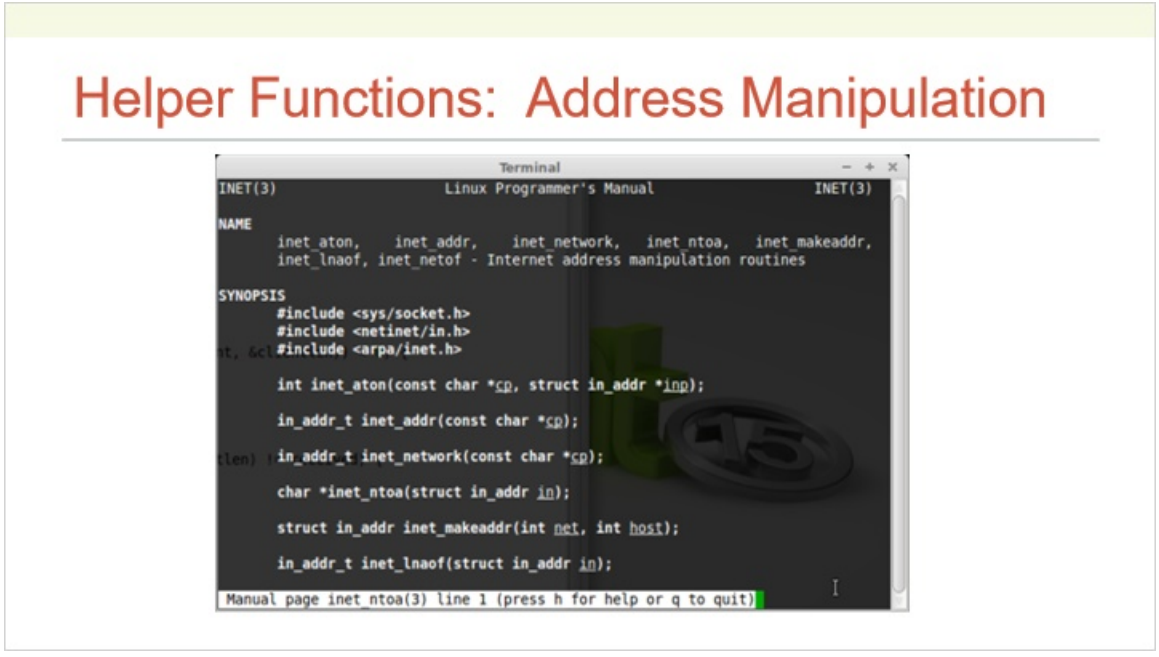
1.15 structure sockaddr_in: members



1.16 Helper Functions: byte order

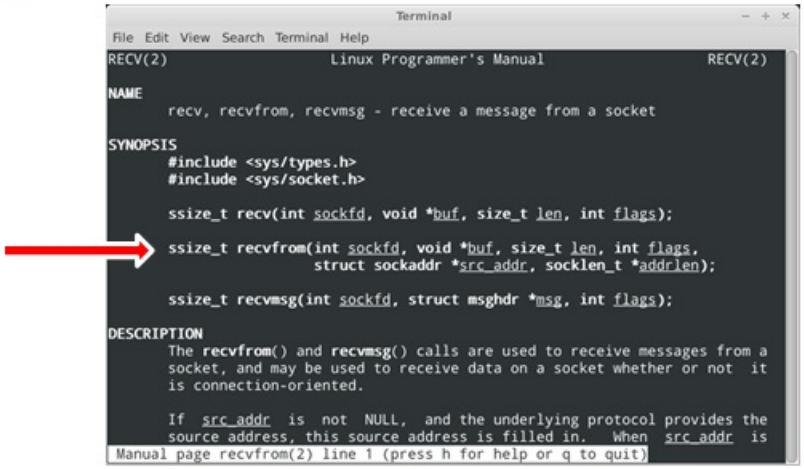


1.17 Helper Functions: Address Manipulation



1.18 System call: recvfrom

System call: recvfrom



1.19 System call: recvfrom

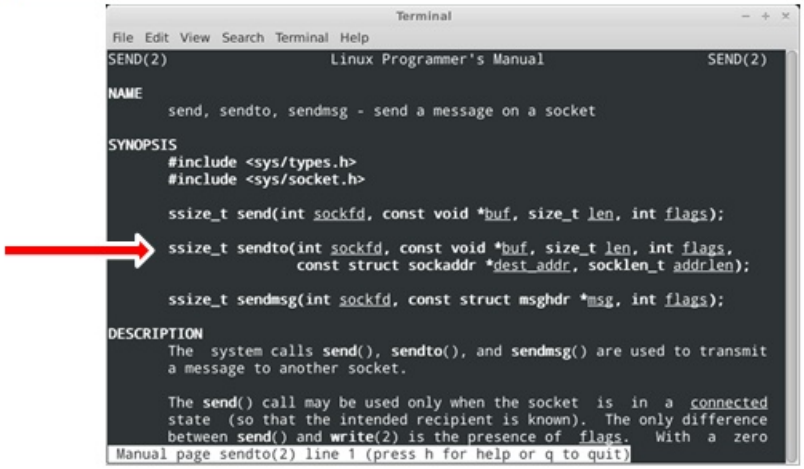
System call: recvfrom

```
ssize_t recvfrom(int sockfd, void *buf, size_t len,
    int flags, struct sockaddr *src_addr,
            socklen_t *addrlen)
```

- receives a datagram **buf** of size **len** from socket **sockfd**
  - will wait until a datagram is available
  - **flags** specifies wait behavior, e.g.: 0 for default
- **src_addr** will hold address information of sender
  - **struct sockaddr** defines address structure
  - **addrlen** specifies length of **src_addr** structure
- returns the number of bytes received, i.e. size of datagram

1.20 System call: sendto

System call: sendto



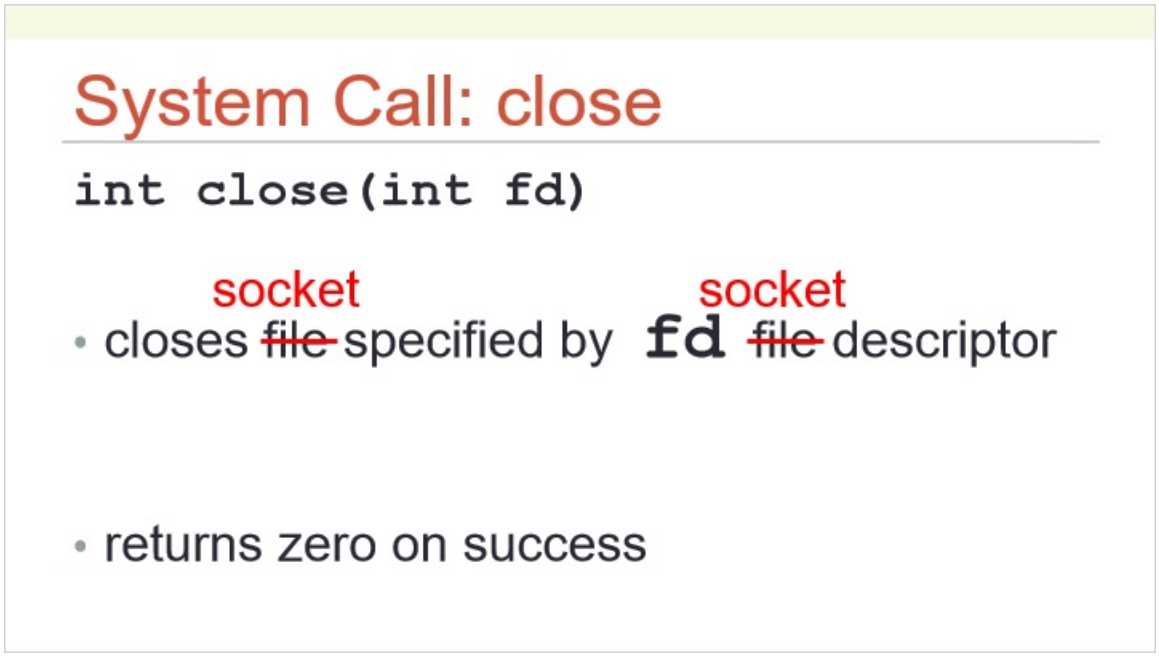
1.21 System call: sendto

System call: sendto

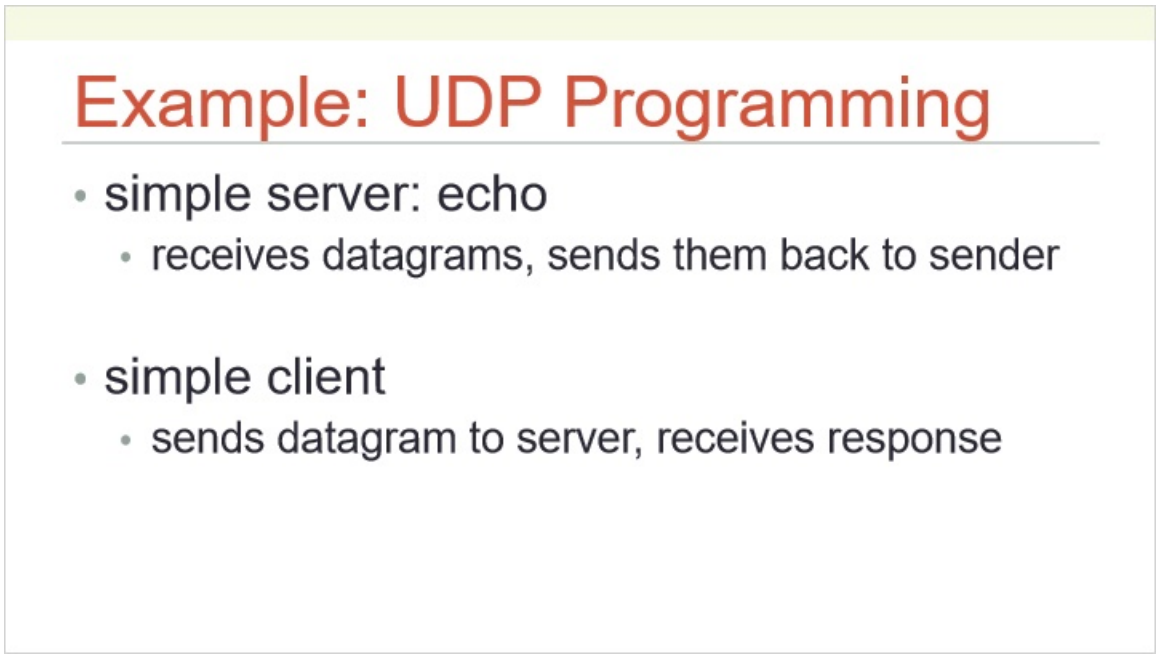
```
ssize_t sendto(int sockfd,
  const void *buf, size_t len, int flags,
  const struct sockaddr *dest_addr, socklen_t addrlen)
```

- sends datagram **buf** of size **len** to socket **sockfd**
  - will wait if there is no ready receiver
  - flags specifies wait behavior, e.g.: 0 for default
- **dest_addr** holds address information of receiver
  - **struct sockaddr** defines address structure
  - **addrlen** specifies length of **dest_addr** structure
- returns the number of bytes sent, i.e. size of datagram

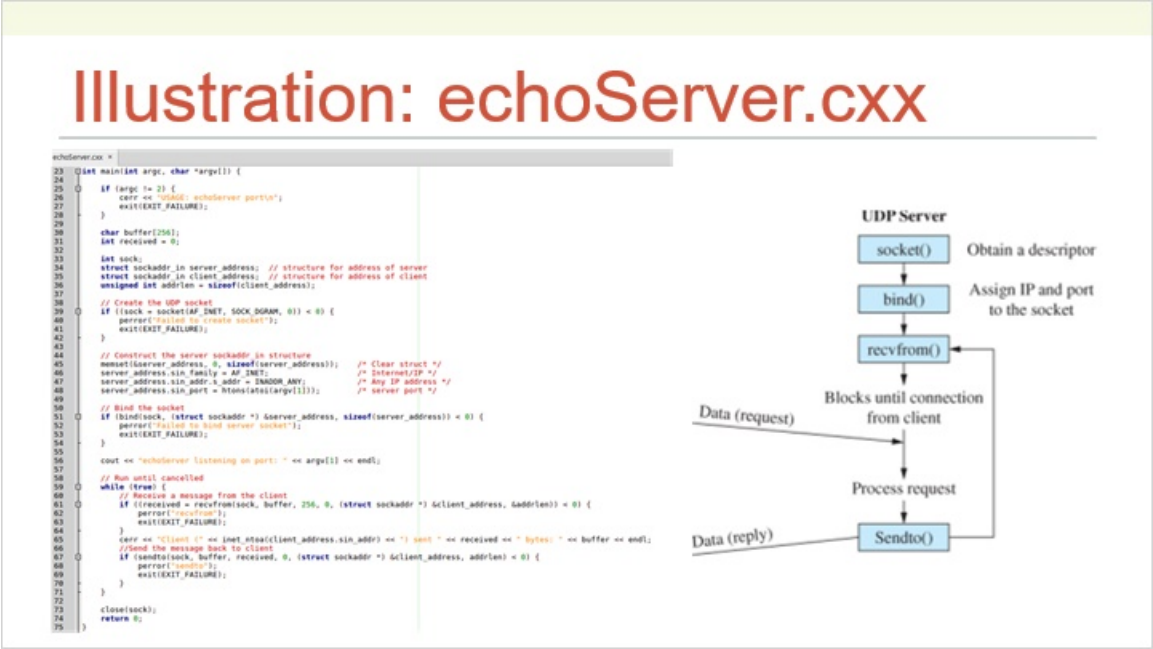
1.22 System Call: close



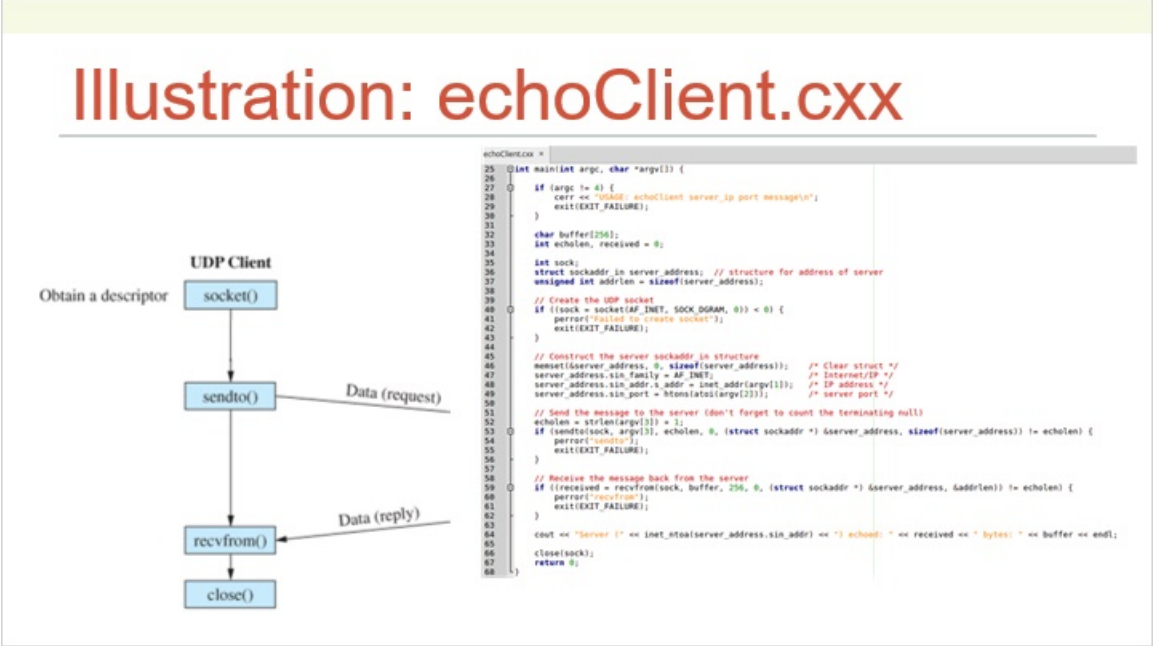
1.23 Example: UDP Programming



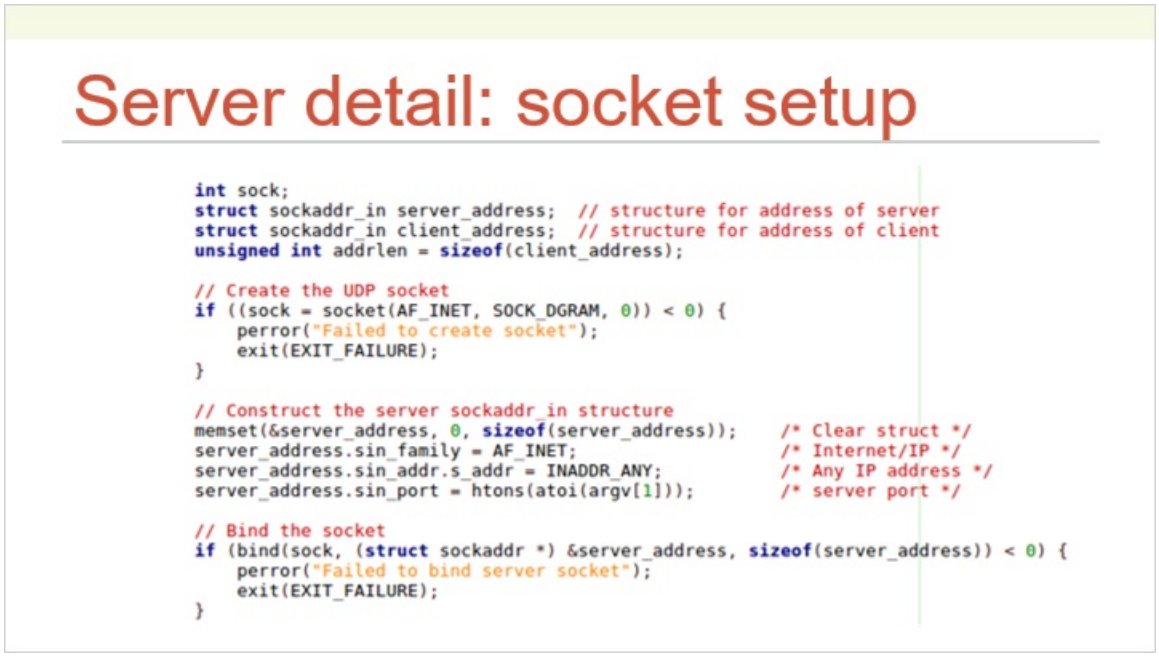
1.24 Illustration: echoServer.cxx



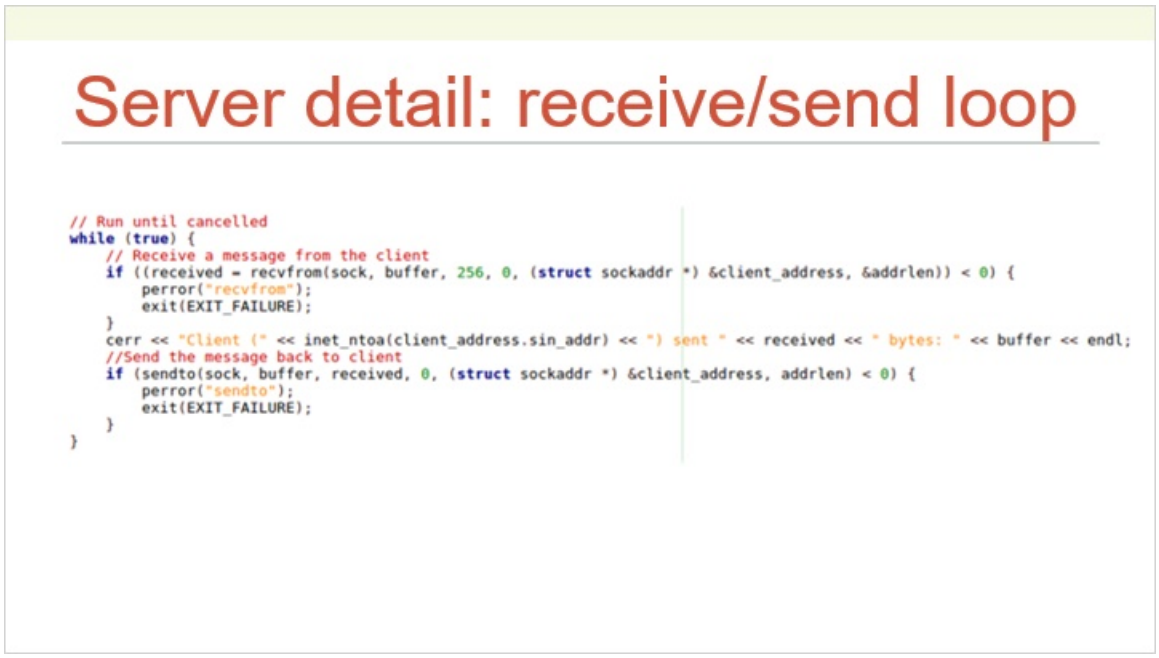
1.25 Illustration: echoClient.cxx



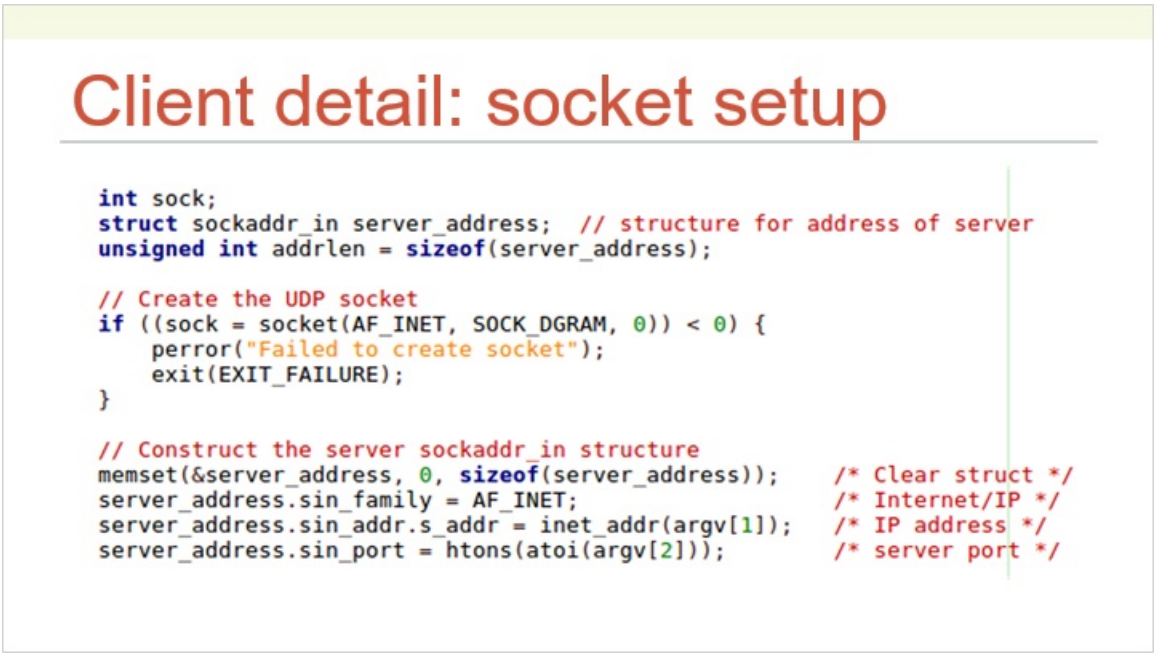
1.26 Server detail: socket setup



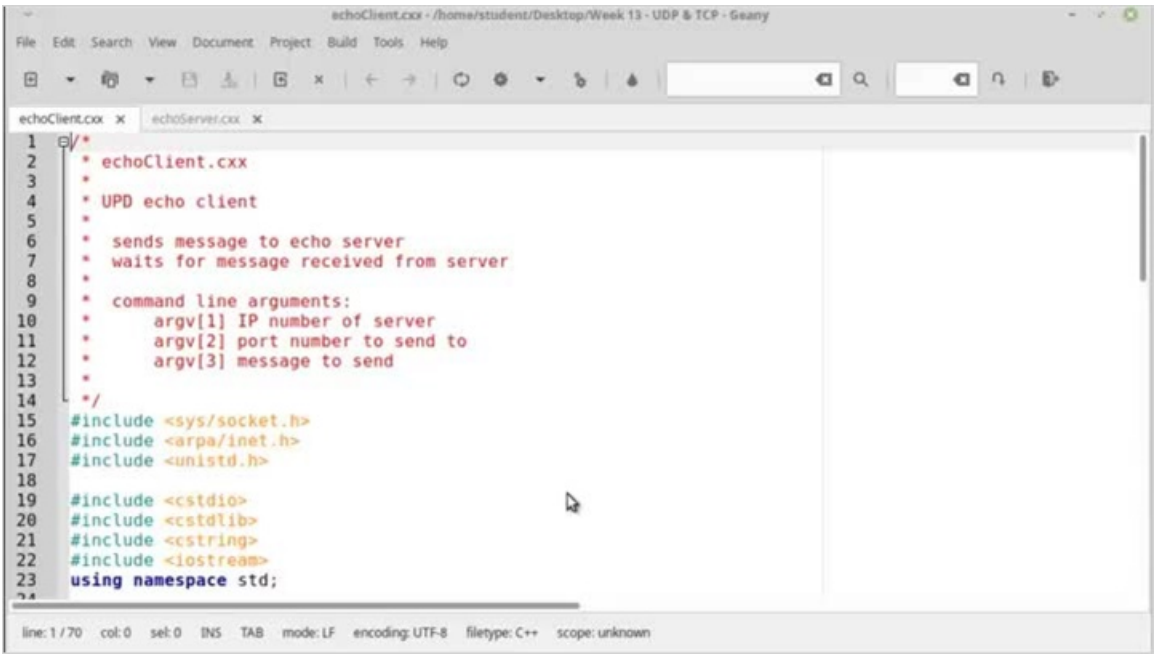
1.27 Server detail: receive/send loop



1.28 Client detail: socket setup



1.29 Echo Example



1.30 Summary


Summary
• Transport layer
• User datagram protocol
• UDP programming
• Example UDP client/server programs