UDP Socket Programming

- Reading data
- Creating UDP sockets
- Sending data
- Receiving data
- Writing data
Reading data

```c
int open(const char *pathname,
         int flags,
         mode_t mode);

int read(int fd,
         char *buf,
         size_t count);
```
Creating a UDP socket

```c
int socket(int family, int type, int proto);

int sock;
sock = socket(PF_INET,
              SOCK_DGRAM,
              0);
if (sock<0) { /* ERROR */ }
```
Sending UDP Datagram

ssize_t sendto(int sockfd, void *buff, size_t nbytes, int flags, const struct sockaddr* to, socklen_t addrlen);

sockfd is a UDP socket
buff is the address of the data (nbytes long)
to is the address of a sockaddr containing the destination address.
Return value is the number of bytes sent, or -1 on error.
The return value of `sendto()` indicates how much data was accepted by the OS for sending as a datagram – not how much data made it to the destination.
Receiving UDP Datagram

```c
ssize_t recvfrom(
  int sockfd,
  void *buff,
  size_t nbytes,
  int flags,
  struct sockaddr* from,
  socklen_t *fromaddrlen);
```

- `sockfd` is a UDP socket
- `buff` is the address of a buffer (nbytes long)
- `from` is the address of a sockaddr.

Return value is the number of bytes received and put into buff, or -1 on error.
recvfrom() 

- If `buff` is not large enough, any extra data is lost.
- You can receive 0 bytes of data.
- The `sockaddr` at `from` is filled in with the address of the sender.
- You should set `fromaddrlen` before calling.
- if `from` and `fromaddrlen` are NULL we don't find out who sent the data.
Write data

```c
int open(const char *pathname,
         int flags,
         mode_t mode);

size_t write( int fd,
              const char *buf,
              size_t count);
```
Don't forget to close()

```c
int close(int fd);
```