Sockets

44-550: Operating Systems

Sockets

- Send data over a network interface
  - This is *not* a networking class
  - We will focus mainly on the use of sockets, not the implementation, UDP vs. TCP, etc, etc, etc...
- Different socket implementations
  - Berkeley sockets (*NIX, most every normal thing)
    - Became the POSIX socket API
    - Also called BSD sockets
    - Just let these terms be interchangable
  - Winsock (or Windows Socket API/WSA) (Windows)
- While we will focus on the BSD socket API, know that there exist some nice abstractions away from these sockets
  - ZeroMQ
  - NanoMsg
  - ...

Two Sides

- “Server”
  - Creates a socket
  - *Binds* to a port
  - *Listens* on that address
  - *Receives* or *sends* data along the socket
- “Client”
  - *Creates* a socket
  - Contacts the server with two arguments: *host* and *port*
  - *Sends* or *receives* information from the server

The Sockets API

- `#include <sys/socket.h>`
  - Also helpful to `#include<sys/types.h>`
- Sockets are represented as ints (handles to sockets)
- Provides functionality to:
  - create a socket
  - bind socket to a port
  - listen on the socket
  - accept connections
  - read/write to/from sockets
  - close the socket

Creating a Socket

Socket Creation (Server and Client)

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

Returns the integer socket handle, -1 if error.

- Lots of different domains
  - We will focus on the `AF_INET` domain: IPv4
  - Other interesting types include `AF_UNIX`/`AF_LOCAL` and `AF_INET6`
- Several different types:
  - `SOCK_STREAM` is TCP, `SOCK_DGRAM` is UDP.
  - Still others
  - See the socket man page
- We will not cover the protocol flag; it is specific to the socket types. Set it to zero for the purposes of this class.

Binding a Socket

Socket Binding (Server)

```
struct sockaddr {
    sa_family_t sa_family;
    char sa_data[14];
};

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

Returns 0 on success, -1 on error

There are easier ways than using a raw `sockaddr` we will discuss when showing an example.

Listining for Incoming Connections, and Accepting

Listen for Incoming Connections (Server)

```
int listen(int sockfd, int backlog);
```

Returns 0 on success, -1 on error

Accept Incoming Connections (Server)

```
int accept(int sockfd, struct sockaddr *addr,
        socklen_t * addrlen);
```

Waits for a socket to try to connect. Sets the sockaddr pointer to the information about the incoming connection. Returns a nonnegative integer (socket descriptor) on success, -1 on error. This will create a new socket for the connection.

Connecting a Client

Socket Connection (Client)

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

Connects the socket to the address specified by `addr`. Returns -1 on error and 0 on success.

Sending and Receiving Data

Send Data (Server and Client)

```
size_t send (int sockfd, const void * buf, size_t len,
       int flags);
size_t write(int sockfd, const void * buf, size_t len);
```

Sends `len` bytes to the connected socket specified by `sockfd`. Returns -1 on a locally defined error, else number of bytes sent. When flags is 0, send and write behave identically.

Read Data (Server and Client)

```
size_t recv(int sockfd, void * buf, size_t len, int flags);
```

Gets maximum of `len` bytes from the socket, and stores them in the memory pointed at by `buf`, Returns -1 on error or number of bytes read.