Client/Server Paradigm

- Chapter 3
- TCP/IP Sockets in C: Practical Guide for Programmers
  - Chapter 1-4

Based on notes by Jim Martin

Client/Server Paradigm

- Server (any program that offers a service that can be reached over a network):
  - Open a port
  - Wait for a client
  - Start slave
  - Continue
- Client (a program that sends a request to a server and waits for the response):
  - Open a port
  - Connect with a server
  - Interact with the sever
  - Close

Based on notes by Jim Martin
Client/Server Paradigm

- Concepts to be aware of:
  - Standard vs nonStandard services (e.g., ftp vs UDPEchoServer)
  - Connectionless vs connection-oriented
  - Stateless vs stateful
  - Concurrent vs iterative (server)
    - Concurrent: if the server handles multiple requests concurrently
    - Iterative: “one request at a time”.

TCP Client / Server Interaction

<table>
<thead>
<tr>
<th>TCP Client</th>
<th>TCP Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>socket()</td>
<td>socket()</td>
</tr>
<tr>
<td>bind()</td>
<td>bind()</td>
</tr>
<tr>
<td>listen()</td>
<td>listen()</td>
</tr>
<tr>
<td>accept()</td>
<td>accept()</td>
</tr>
<tr>
<td>socket()</td>
<td></td>
</tr>
<tr>
<td>connect()</td>
<td></td>
</tr>
<tr>
<td>write()</td>
<td>read()</td>
</tr>
<tr>
<td>read()</td>
<td>write()</td>
</tr>
<tr>
<td>close()</td>
<td>read()</td>
</tr>
<tr>
<td></td>
<td>close()</td>
</tr>
</tbody>
</table>

Based on notes by Jim Martin
TCP Socket Calls

- int connect(int sockfd, const struct sockaddr *dstaddr, int addrlen)
  - used by TCP to establish a connection
- ssize_t read(int sockfd, void *buf, size_t nbytes)
- ssize_t write(int sockfd, void *buf, size_t nbytes)

TCP Socket Calls

- Listen: called only by a TCP server.
  - Converts an active socket into a passive socket meaning the kernel should accept incoming connection requests.
  - Sets the maximum number of connections the kernel should queue for this socket.

- int listen (int sockfd, int backlog)
  - There are two queues:
    - incompled cx queue
    - completed cx queue
  - the backlog (roughly) indicates the sum of the two queues
TCP Socket Calls

- **Accept**: Called only by a TCP server to return the next completed connection from the completed queue.
- ```c
int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addren)
```
- Returns a new socket descriptor. Thus a server will have a listenfd and a connectfd.

Send socket call

- ```c
int send(int socket, const void* msg, unsigned int msgLength, int flags)
```
- `socket`: must be connected!!
- `msg`: ptr to data
- `msgLength`: # bytes to send
- `flags`: control flags (0 unless you know what you are doing!!)

- Returns number of bytes sent, -1 otherwise
- If the msgLength exceeds socket send buffer size, the process blocks until all data can be accepted by TCP.
**Recv() socket call**

- `int recv(int socket, const void* rcvBuffer, unsigned int bufferLength, int flags)`
  - `socket`: must be connected!!
  - `msg`: ptr to where data is to go
  - `bufferLength`: max number of bytes to rx
  - `flags`: control flags (0 unless you know what you are doing!!)

- Returns number of bytes received, -1 on error, a 1 if the other side disconnects.
- Note: you might have to loop until all bufferLength bytes arrive

Based on notes by Jim Martin