**EECS 338**

***Today...***

1. **Pipes**
2. **Sockets**

**Announcements**

* Tuesday (recitation):
  + Review of HW #4 concepts
  + HW help
  + Pickup graded quizzes
* Wednesday: multithreading (Ch 4)
* Friday: **NO CLASS**
  + Chris will be here: <https://sigcse2018.sigcse.org/>

**IPC: pipes**

* Within a program: only communicates between parent and child processes
* Like a unidirectional mailbox
* Parent and children can all write and read.
* unix\_pipe.c (from textbook)
* pipe\_double.c (from Chris)
* <https://en.wikipedia.org/wiki/File_descriptor>
* Different: pipes in the shell (command line)

**IPC: Sockets**

* Communication between unrelated processes (don’t need to use fork())
* Even on different computers and networks
* What does TCP/IP stand for?

**IPC: Sockets**

Transmission Control Protocol (TCP)

Internet Protocol (IP)

TCP is responsible for the data delivery of a packet and IP is responsible for the logical addressing. In other words, IP obtains the address and TCP guarantees delivery of data to that address. (Source: <http://searchnetworking.techtarget.com/>)

The first two-network TCP/IP transmission was between the van and ARPANET. See <https://en.wikipedia.org/wiki/Packet_Radio_Van>.

**IPC: Sockets**

* C example
  + <http://www.linuxhowtos.org/C_C++/socket.htm>
  + server.c, client.c
  + Note server and client both read and write.
  + Usage
    - ./server 1000 &
    - ./client 127.0.0.1 1000

**Ch 4: Threads**

thread: <https://en.wikipedia.org/wiki/Thread_(computing)>

|  |  |
| --- | --- |
| The smallest sequence of programmed instructions that can be managed independently by a scheduler. In most cases a thread is a component of a process. Multiple threads can exist within one process and share resources such as memory, while different processes do not share these resources. In particular, the threads of a process share its instructions (executable code) and its context (the values of its variables at any given time). |  |