Today's class

- Will continue from last class on IPC mechanisms
- Will go over some programs that demonstrate the various IPC mechanisms
- Will introduce threads

Learning objectives
- To have an understanding of the behavior of basic IPC system calls
- Behavior of blocking vs non-blocking
- Basic, high-level differences between processes and threads
Pipe example on blocking I/O
Message-Passing IPC

• Synchronous vs Asynchronous
  – Synchronous: Sending and reception are practically/logically considered to happen at the same time.
    • This is done by blocking send or blocking receive
  – Asynchronous: sending and reception are done at separate times.
    • Direct asynchronous (uncommon)
    • Indirect asynchronous (common)

• Symmetry vs asymmetry
  – Symmetry: both receiver and sender name each other
  – Asymmetry: only one needs to name each other
    • What about the case that noone names anyone?
Sockets: a message-passing mechanism

- `int socket(AF_UNIX, type, 0);`
- `int socketpair(AF_UNIX, type, 0, int *sv);`
- `int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);`
- `int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);`
Threads
What are threads?

- Threads are also called **lightweight processes**
- Conventional processes are called **heavyweight processes**
- Some definitions:
  - Threads are “mini” processes that run within the memory space of one process.
  - Threads are independent, concurrent *instruction execution paths* within one process
Threads will dynamically use up stack and heap space. Every new allocation (of new data or new function call) will be attributed to the calling thread, so threads have their own heap and stack, but all are within the same address space!
(Heavyweight) Processes vs Threads

• Processes
  – Separate memory space
  – They are visible with OS utilities, e.g. `ps` and `top`.

• Threads
  – Common memory space
    • Communication of data among threads is more convenient
  – Faster to create (20×) and switch (6×) than processes
Next class

- More on threads and their implementation
  - Sections 4.1–4.3