Interprocess Communication

1. **Buffering**	Queue of messages attached to the link Three ways: > zero capacity: no messages are queued on a link sender must wait for the receiver > bounded capacity: finite length of n messages >unbounded: infinite length
2. **Communications in client-server systems**	Sockets > endpoint for communcation (161.25.19.8:1625) Remote Procedure calls > used between systems with network connections pipes > allows two processes to communicate Unix signals >used to notify the process of an event
3. **Indirect Communication (properties of communication link)**	Messages are directed to and received from mailboxes > each mailbox has a unique id Properties > link is established only if processes share a common mailbox > A link may be associated with many processes each pair of processes may share several communication links > may be unidirectional or bi-directional

4. **Message passing (2 steps, implementation)**	Processes communicate with each other without restoring to shared variables.
	> if Processes p and q wish to communicate they need to :
	>>> establish a **Communication link** between them
	>>> exchange messages via send/receive
	
	Implementation
	> physical: memory hardware bus, network
	> logical: direct or indirect, synchronous or asynchronous, automatic or explicit buffering

5. **Naming: Direct communication (properties)**	Must name each other exylicitly: > send(P, Message); send message to p **Properties** > links are established automatically > a link is associated to **one** pair of communicating processes > between each pair there exists exactly one link > the link may be unidirectional, but usually bi-directional
6. **Producer Consumer buffers**	Unbounded: no practical limit on the size of the buffer Bounded: assumes that there is a fixed buffer size
7. **Shared Memory**	> Area of memory shored among the processes that wish to communicate > Communication is under the control of the users processes not the OS
8. **Synchronizaton**	Message passing may be either blocking or non blocking > **blocking is synchronous** >>> sending or receiving blocks > **non-blocking is considered asynchronous**
9. **Two models of IPC**	Shared Memory Message Passing
10. **Types of Pipes**	**Ordinary** Allows communication in standard producer consumer style > producer writes to one end, consumer reads from the other end > requires parent-child relationship **Named** > no parent-child relationship needed > several processes can use the same named pipe