Types of IPC's

1. Unnamed Pipes - Unnamed pipes are a byte stream buffer in the kernel that sends data one-way. One process can write to it but, but can't read from it, and another process can read from it, but can't write to it. Because they are unnamed, they do not take up space in the file system, but do have a limited capacity. Pipes are best used for sending small amounts of data without needing to receive data back.
2. FIFO’s (Named Pipes) – FIFO’s are similar to pipes, with the exception that they are named files in the filesystem. Because of this, any process can open and use a given FIFO. FIFO’s are best used for sharing data between any number of processes, but not at the same time.
3. POSIX Message Queues – Similarly to FIFO’s, POSIX Message Queues uses a named file in the file system to store messages. Unlike FIFO’s, however, Message Queues can be written do and read by multiple processes at a time. Message Queues also supports message priority to determine a messages position in the queue and can notify a process if a new message has arrived in an empty queue. Message Queues’ are best for sending and receiving data between multiple processes simultaneously.
4. Shared Memory – Like the name implies, Shared Memory is when multiple processes share access to the same section of memory. This is a very efficient method of IPC, at the cost of security. There are multiple ways of using Shared Memory:
   1. Shared Anonymous Mapping – Memory is shared between related processes which allocate a zero-initialized block of memory of a specified length. Shared Anonymous Mapping is best used when shared between a parent processes and it’s child process.
   2. Shared File Mapping – Memory is shared between unrelated processes, which allocate a block of memory that is initialized from a file. Any updates to the shared memory are also applied to the initial file. Shared File Mapping is best used between separately run processes whose shared data needs to be saved.
   3. POSIX Shared Memory – Memory is shared between unrelated processes, without using a shared file, thus reducing file I/O overhead. POSIX Shared Memory is best used between separately run processes whose shared data does not need to be remembered.
5. POSIX Semaphores – POSIX Semaphores represents a some amount of shared resources. Any number of processes can take resources from the Semaphore, but if it runs out, than any new processes must wait until one of the older processes is done with their portion of the resources. Semaphores are best used for enforcing order of instructions between processes and preventing race conditions.. There are two types of Semaphores:
   1. Unnamed Semaphores – Stored in shared memory.
   2. Named Semaphores – Stored as a named object.
6. Sockets – Sockets act as endpoints between two processes, thus each process needs it’s own Socket. Because of this, communication between sockets are bidirectional. Each Socket has a domain which determines how data is transferred between processes, the address to identify other Sockets with, and which processes can share data. There are three common types of domains:
   1. UNIX Domain – All communication is between processes on a single host through a temporary file in the filesystem, whose path acts as the address.
   2. IPv4 Domain – Communication is between processes on different hosts on an IPv4 network using an IPv4 address and port umber as the address.
   3. IPv6 Domain - Communication is between processes on different hosts on an IPv6 network using an IPv6 address and port umber as the address.

   There are three main types of sockets:

1. Stream Socket – Data is transferred through a byte stream which sends the data in it's entirety or not at all. Before transfer, a connection is established between the two Sockets to ensure that the data arrives reliably and intact. If over the internet, Stream Sockets use the TCP protocol.
2. Datagram Socket – Data is sent as multiple messages which can arrive multiple times, out of order, or even not at all. This is because Datagram Sockets do not bother with establishing a connection or making sure that the message got sent, instead opting to send the message out and not worry about anything that happens after. In the UNIX domain, Datagram Sockets are reliable and also allow for broadcasting datagrams to multiple destinations. If over the internet, Datagram Sockets use the UDP protocol.
3. Sequential Packet Sockets – This socket type acts as a blend between Stream Sockets and Datagram Sockets. Sequential Packet Sockets establish a connection between the two sockets, thus ensuring reliability, but send the data as multiple messages. This Socket type is provided by Unix and Unix-like systems. If over the internet, Sequential Packet Sockets use the SCTP protocol.