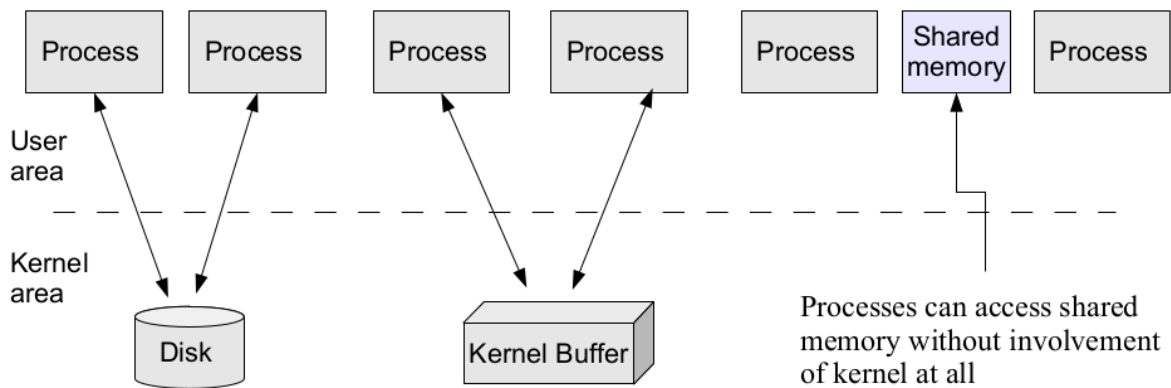# Application Design

Option 1

One huge monolithic program that does every thing

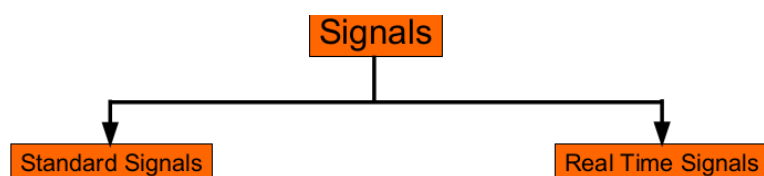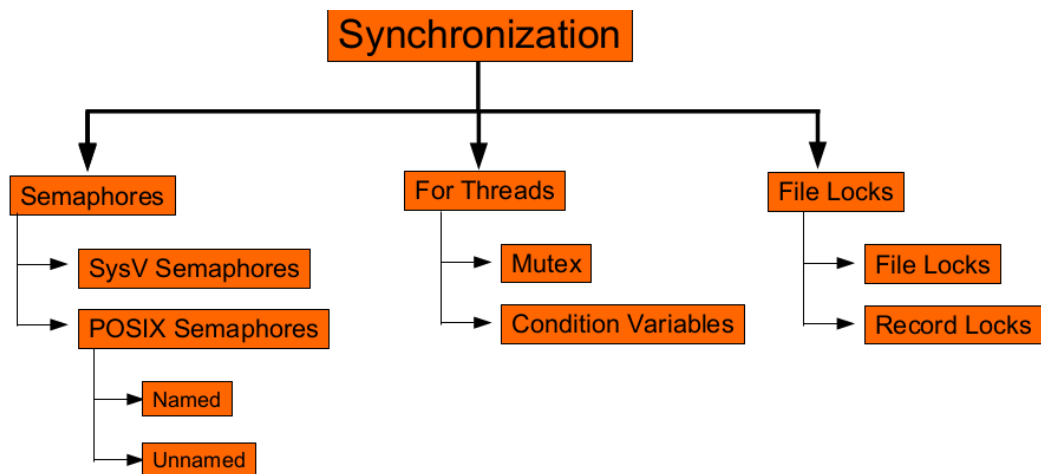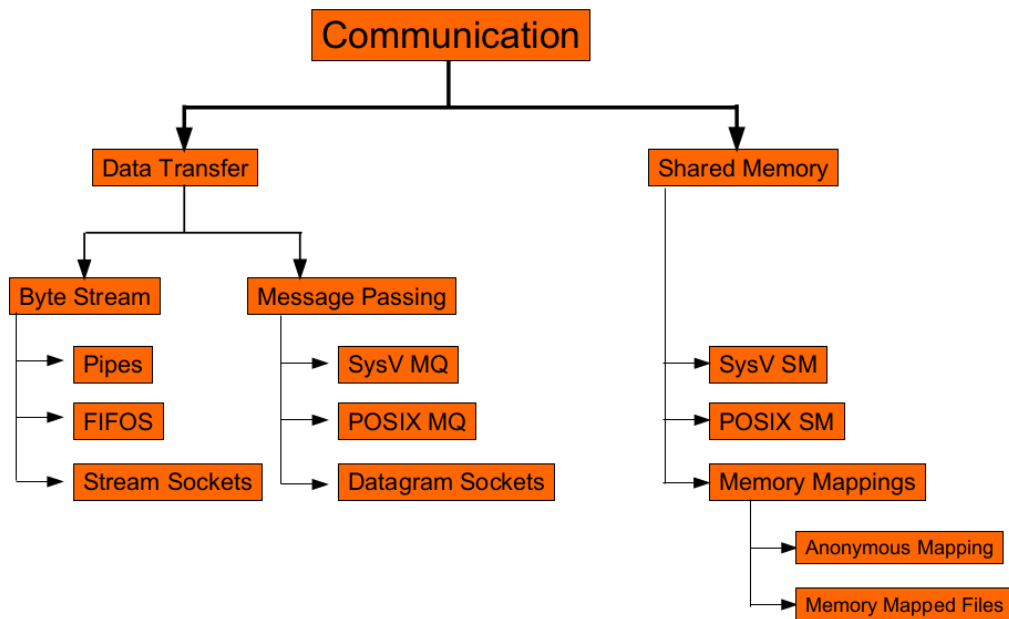## Option 2

Multi_threaded programs

Option 3

Multiple programs using `fork()` that communicate with each other using some form of Inter Process Communication (IPC)

Ways to Share Information b/w UNIX Processes

Processes can access shared memory without involvement of kernel at all

Categories of IPC

- Communication
- Synchronization
- Signals

Communication

- Data Transfer
  - Byte Stream
    - Pipes
    - FIFOS
    - Stream Sockets
  - Message Passing
    - SysV MQ
    - POSIX MQ
    - Datagram Sockets
- Shared Memory
  - SysV SM
  - POSIX SM
  - Memory Mappings
    - Anonymous Mapping
    - Memory Mapped Files

Synchronization

- Semaphores
  - SysV Semaphores
  - POSIX Semaphores
    - Named
    - Unnamed
- For Threads
  - Mutex
  - Condition Variables
- File Locks
  - File Locks
  - Record Locks

Signals

- Standard Signals
- Real Time Signals

♔ Persistence of IPC objects

Process Persistence
- Exists as long as it is held open by a process
- Pipes and FIFOs
- TCP, UDP sockets
- Mutex, condition variables, read write locks
- POSIX memory based semaphores

Kernel Persistence
- Exists until kernel reboots or IPC objects is explicitly deleted
- Message Queues, semaphores & shared memory are at least kernel persistent

File system Persistence
- Exists until IPC objects is explicitly deleted, or file system crashes
- Message queues, semaphores & shared memory can be file system persistent if implemented using mapped files