Inter-Process Communication (IPC)

- Historically only for disjoint, single-threaded processes:
  - Data channels: pipes, sockets, messages
  - Asynchronous signals
- For modern operating systems, threading and sharing must be considered:
  - Synchronization mechanisms: Semaphores, mutexes, ...
  - Means of sharing address space
- Separate mechanisms for intra and inter process synchronization?

Unix IPC

- Process-oriented — later adapted to threading

Traditional

- Pipes, named pipes (FIFOs)
- Signals

System V IPC

- Semaphores (atomic multi-wait)
- Message queues
- Shared memory

Other

- Sockets
- Pthread mutexes/conditions *may* be provided across processes
- POSIX semaphores *may* be provided across processes

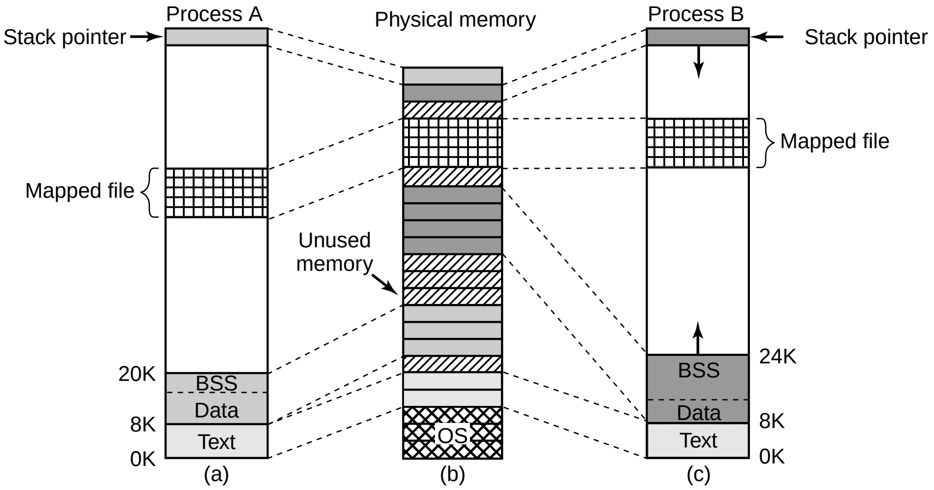
Unix Signals

- Signals can be considered *software interrupts*
- Fixed number of primitive *signal types* (e.g. `SIGTERM`, `SIGSEGV`)
- Signals are *sent* to processes to notify about user og OS events
- Signals are *handled* in context of process (using current stack)
- Signals handled by default or user-provided *handler functions*
- Signals are handled *asynchronously* unless *blocked*
- Blocked signals may be handled *synchronously* by explicit wait

Observation

- Signals and threads do no blend very well

Shared Memory



Windows IPC

- Designed for multi-threading

Unix-like

- Pipes, named pipes
- Asynchronous Procedure Call (ACP) — executed at wait points
- Semaphores, events, mutexes (no conditions!)
- Mailslots (also with broadcast)
- Shared memory
- Sockets (WinSock)

Specialized

- Clipboard
- Common Object Model (COM) — client/server call model