IPC

- Global variable before the fork
- exit
- signals
- Pipes
- FIFO
- Sockets
- Shared memory
- Text file

Global variable before the fork

- Implementation – kernel based
- Scope - local
- Duplex – no (just one way parent → child)
- Time-coupling - Sim
- Space-coupling – Sim (parent knows child & child knows parent)
- Explicit / implicit - implicit
- Synchronization - yes
- Process relation – parent → child
- Identification – variable names
- API - +- global variable + fork

exit

- Implementation – kernel based
- Scope - Local
- Duplex – no (just child → parent)
- Time-coupling – No
- Space-coupling – yes (parent knows who sent the data) No (child doesn not know if it is the parent pr grand parent)
- Explicit / implicit - explicit
- Synchronization – yes (if we use the wait) no (if we use the signal)
- Process relation – Child → parent
- Identification – no
- API - exit

signals

- Implementation - kernel
- Scope - local
- Duplex - No
- Time-coupling – yes (sender / received exist at the same time)
- Space-coupling – yes (sender knows who receives the signal)
- Explicit / implicit - Explicit
- Synchronization – No
- Process relation – No
- Identification – signal number+process id
- API - Kill

Pipes

- Implementation - kernel
- Scope - local
- Duplex- No
- Time-coupling - yes
- Space-coupling – no (I d not know who is going to receive the data/ I do not know who sent the data)
- Explicit / implicit -Explict
- Synchronization – yes (wait on empty, wait on full)
- Process relation – Parent, Child, siblings
- Identification – fd variables
- API – pipe + file operation

FIFO

- Implementation - kernel
- Scope - local
- Duplex- No
- Time-coupling - yes
- Space-coupling – no (I d not know who is goind to receive the data/ I do not know who sent the data)
- Explicit / implicit -Explicit
- Synchronization – yes (wait on empty, wait on full)
- Process relation – No
- Identification – file name on file system
- API – mkfifo + file opening + file operation

Socket datagram

- Implementation - kernel
-
- Duplex- yes, but application dependent (sendto → recvfrom → sendto)
- Time-coupling - yes
- Space-coupling – yes (need to know the address of the receiver)
- Explicit / implicit -Explicit
- Synchronization – No (send does not wait for the receive) yes (received waits for the sender
- Process relation – No
- Identification – address that depends on the domain
- API – socket + sendto recevfrom

Socket stream

- Implementation - kernel
-
- Duplex- yes connect/accept (send → recv → send) only 2 processes interact
- Time-coupling - yes
- Space-coupling – yes (connect we need to know the address of the server)
- Explicit / implicit -Explicit
- Synchronization – No (send does not wait for the receive) yes (received waits for the sender
- Process relation – No
- Identification – address that depends on the domain
- API – socket + send recev + file API

Socket UNIX

- Implementation - kernel
- Scope - local
-
-
-
- Explicit / implicit -Explicit
-
-
- Identification – address (file name)
-

Socket INET

- Implementation - kernel
- Scope - global
-
-
-
- Explicit / implicit -Explicit
-
-
- Identification – address (IP address + port)
-

Shared memory

- Implementation – shared memory
- Scope - Local
- Duplex - yes
- Time-coupling - No
- Space-coupling - No
- Explicit / implicit - Implicit
- Synchronization – No (requires use of mutex)
- Process relation – no (depending on the API)
- Identification – SysV – numeric id POSIX - name
- API – specific for creatiin – regular memory accesses

FILES

- Implementation – File system
- Scope – Local (if usng local FS) or global
- Duplex - yes
- Time-coupling - No
- Space-coupling - No
- Explicit / implicit - Implicit
- Synchronization – No (requires use of mutex)
- Process relation – no (depending on the API)
- Identification –file name
- API – fopen + file operations