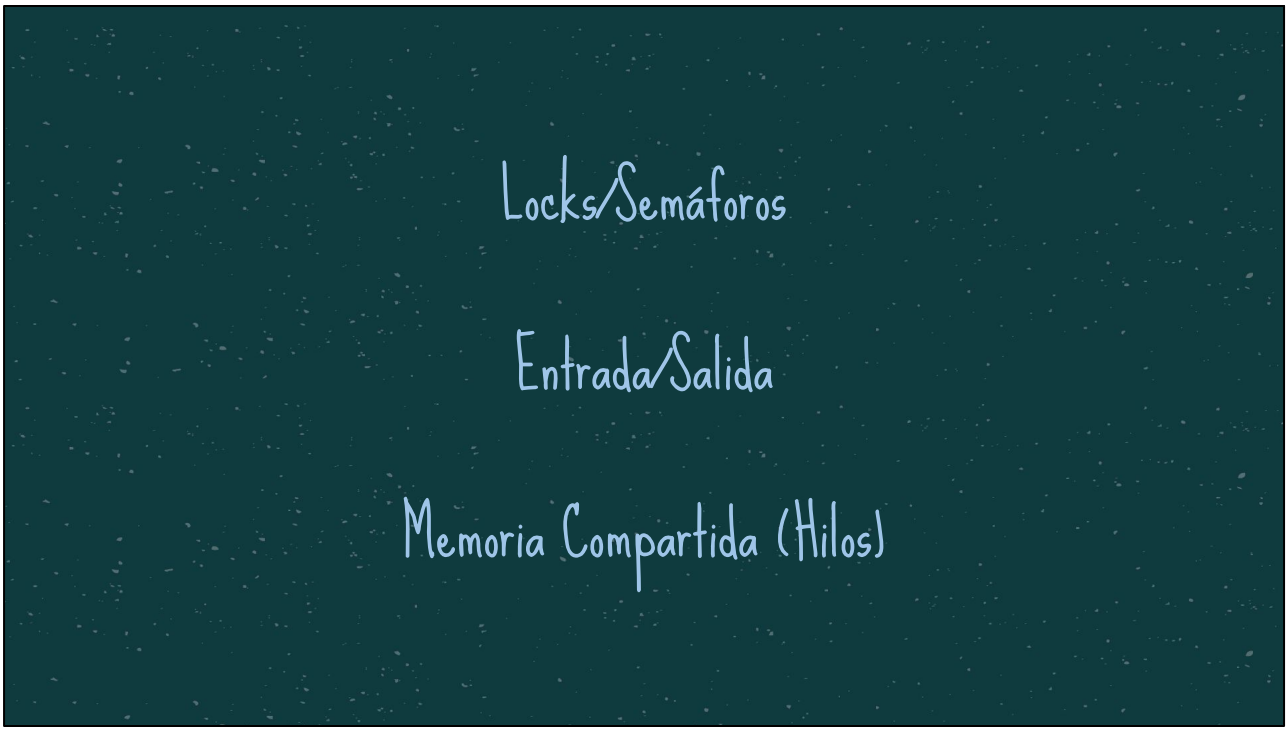
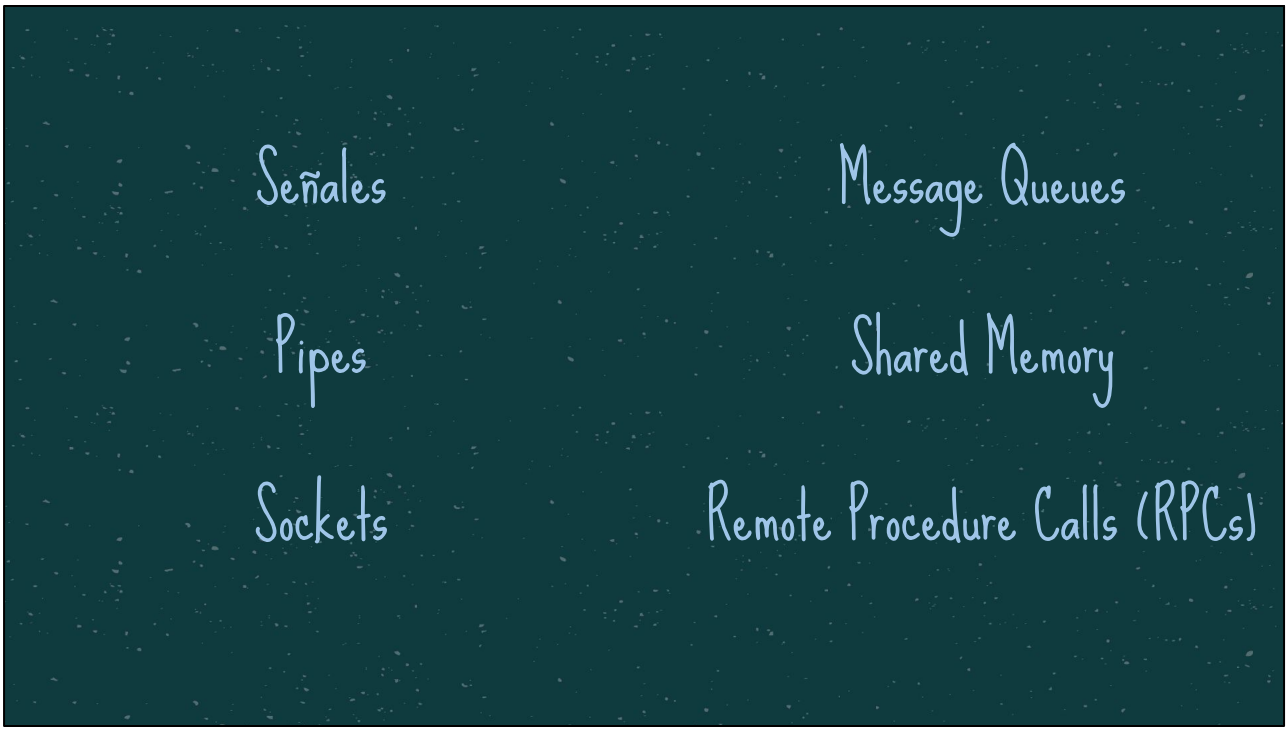
Sistemas Operativos I

Inter-Process Communication (IPC) - El comienzo



Ya vimos algo de comunicación entre procesos/hilos.

+ Locks y semáforos pueden ser utilizados entre procesos y sirven como una forma muy primitiva de comunicación.
+ Entrada y salida (escritura y lectura a un archivo intermedio)
+ Memoria compartida en el caso de hilos (ya que por definición los hilos comparten el espacio de memoria del proceso que los crea).



Pero hay varios mecanismos de comunicación entre procesos que todavía no vimos.

+ Señales: Podemos indicarle al SO que le envíe una señal a un proceso
+ Pipes: tuberías que nos sirve para comunicarnos entre los procesos.
+ Sockets: una generalización de la estructura local para poder comunicarnos entre procesos en por la red.
+ Message Queues : una generalización de Pipes donde los mensajes tiene más estructura (de hecho los mensajes tienen estructura)
+ Shared Memory : memoria compartida entre procesos (no solo entre hilos, librería shm shmget o también mmap para mapear IO/Devices a memoria)
+ Remote Procedure Calls (RPCs) : hacer llamadas a procesos de forma remota, hoy en día se ve mucho con servicios en la internet.

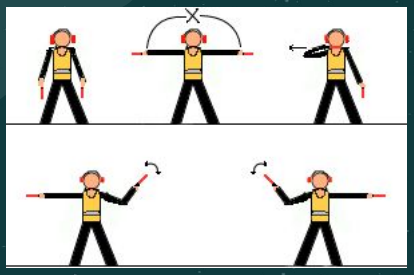
Nosotros por el momento nos concentraremos en Señales/Pipes/Sockets.

+ Message Queues son una generalización de Pipes, en particular tienen : Mensajes con estructura, son bidireccionales, los mensajes se puede recorrer y leer.
+ Shared Memory es memoria compartida entre procesos, ya vimos como era entre hilos y los problemas que traía.

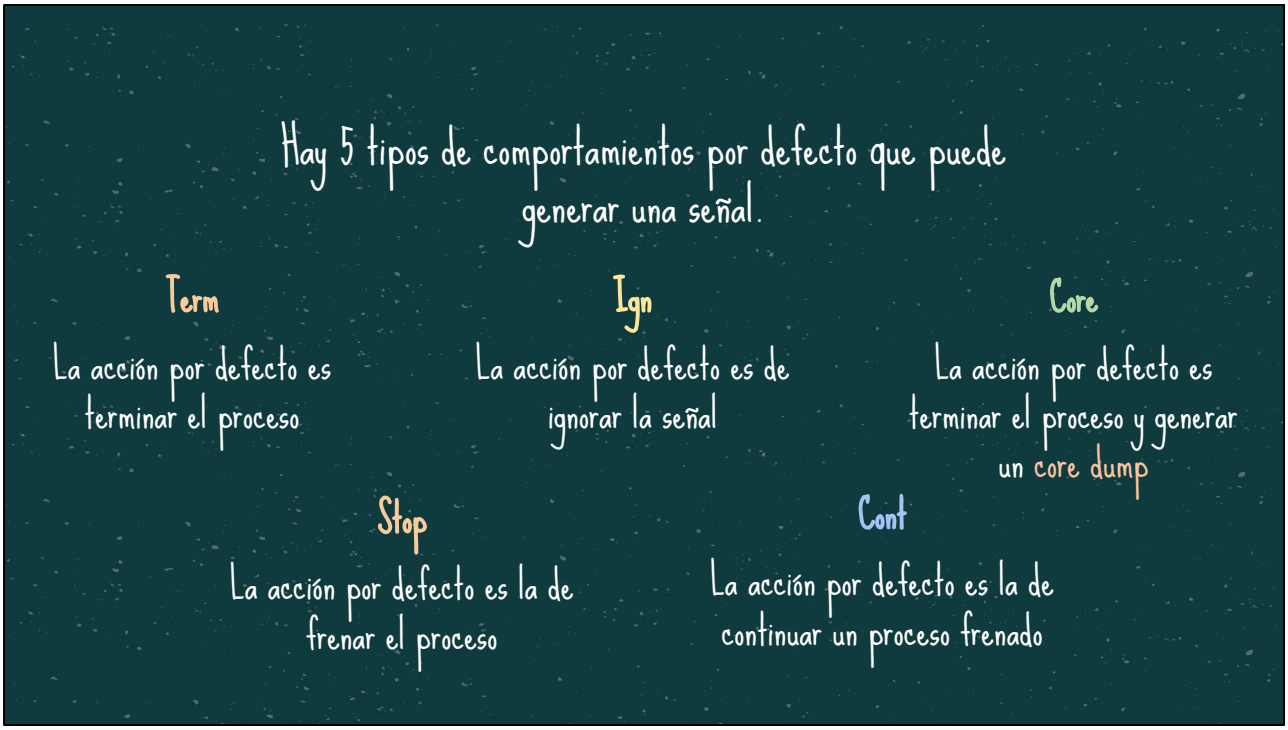
Ambos involucran mayor manejo de C, pero que deberían ser capaces de usarlos por su cuenta

RPCs sobre el final de la materia

Señales



Las señales son interrupciones producidas por un proceso o por el Sistema Operativo (Kernel) y sirven para identificar que algo ha ocurrido.
Por ejemplo, CTRL+C/D, etc, producen señales a los procesos que se están ejecutando en una terminal.
Aunque también se pueden producir por fallos de hardware (objetivo por el cual fueron creadas).
Se dicen que son *generadas* en el momento que se producen y son consideradas *entregadas* cuando el proceso que la recibe actúa sobre ellas.



Más información en `man 7 signal`

Visitamos `man 7 signal` y programamos un manejador de señales `man 3 signal`

Los pipes como indica su nombre introducen un medio de comunicación similar a lo que sería una tubería.
Es un canal *unidireccional* (en una dirección), que permite escribir de un extremo, y leer del otro.
Es básicamente como tener un archivo ya que la creación del pipe nos retornará dos descriptores de archivos, uno para realizar la lectura y otro para realizar la lectura.
Veamos un ejemplo.
Visitamos el manual de pipe: man 3 pipe
Creamos un ejemplo para que se comunique el padre.


Pipes
#include <unistd.h>
int pipe(int fildes[2]);