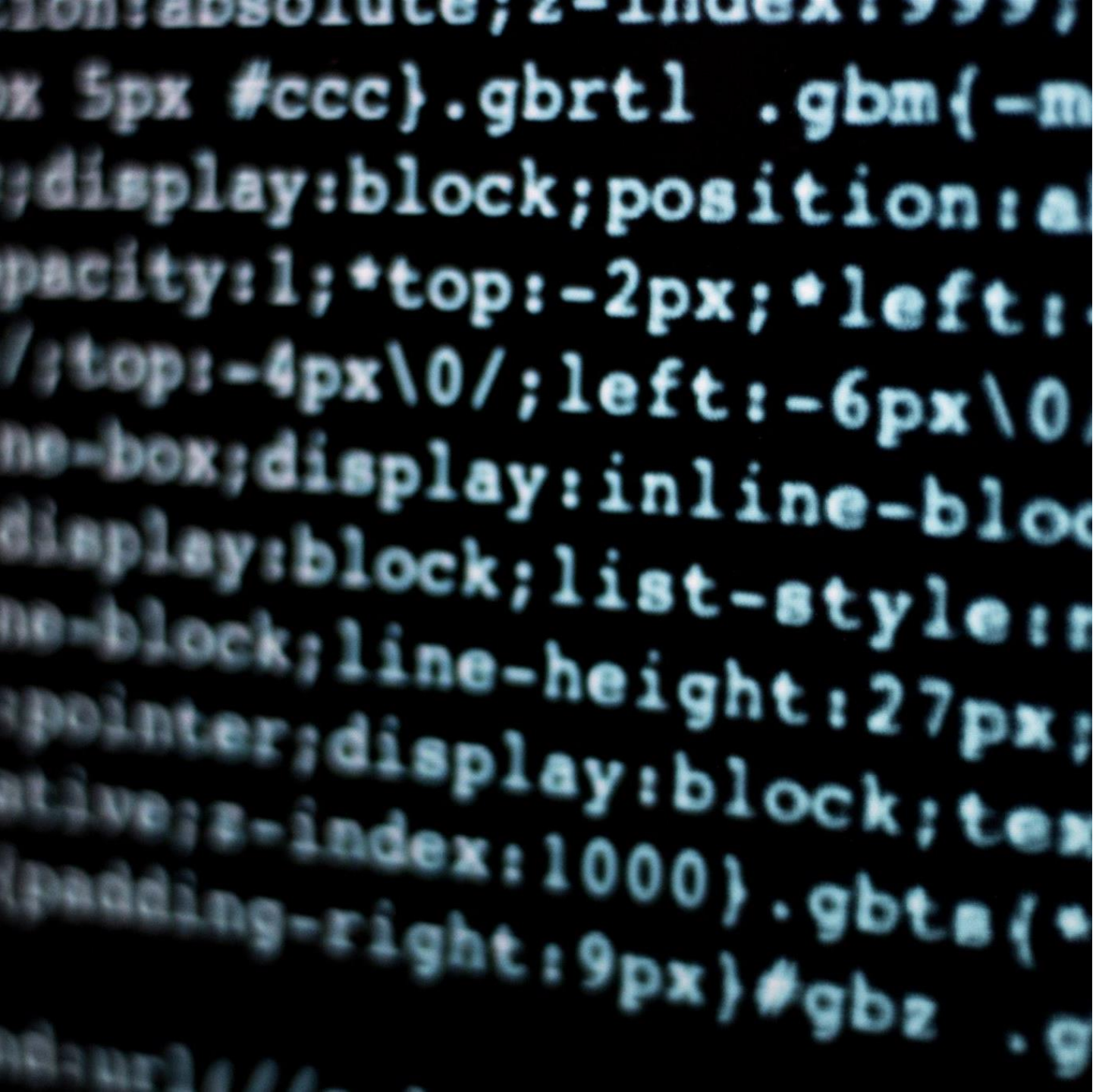
UTN.BA FACULTAD REGIONAL BUENOS AIRES

Arquitectura y Sistemas Operativos

Tecnicatura Universitaria en Programación



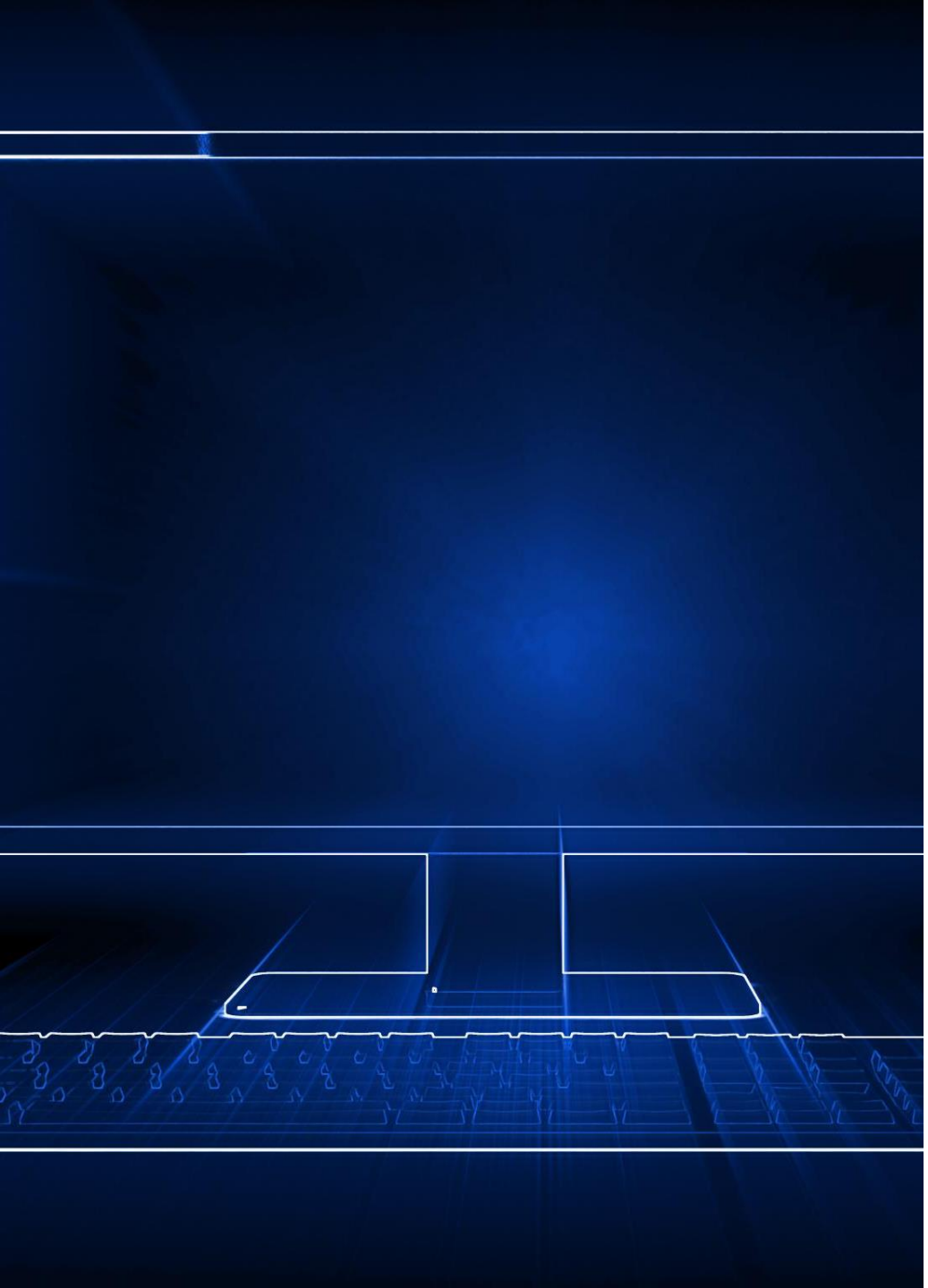
Concurrencia, exclusión mutua y sincronización

Unidad 4

Agenda

1. Principios de concurrencia
2. Interacción entre procesos
3. Exclusión mutua por hardware
4. Exclusión mutua por software

1. Principios de concurrencia



Concurrencia

La concurrencia es fundamental en áreas como la **multiprogramación**, el **multiprocesamiento** y **el procesamiento distribuido**.

La concurrencia abarca aspectos como la comunicación entre procesos, la compartición o competencia por los recursos, la sincronización de actividades y la reserva de tiempo del procesador.

¿Qué es la concurrencia?

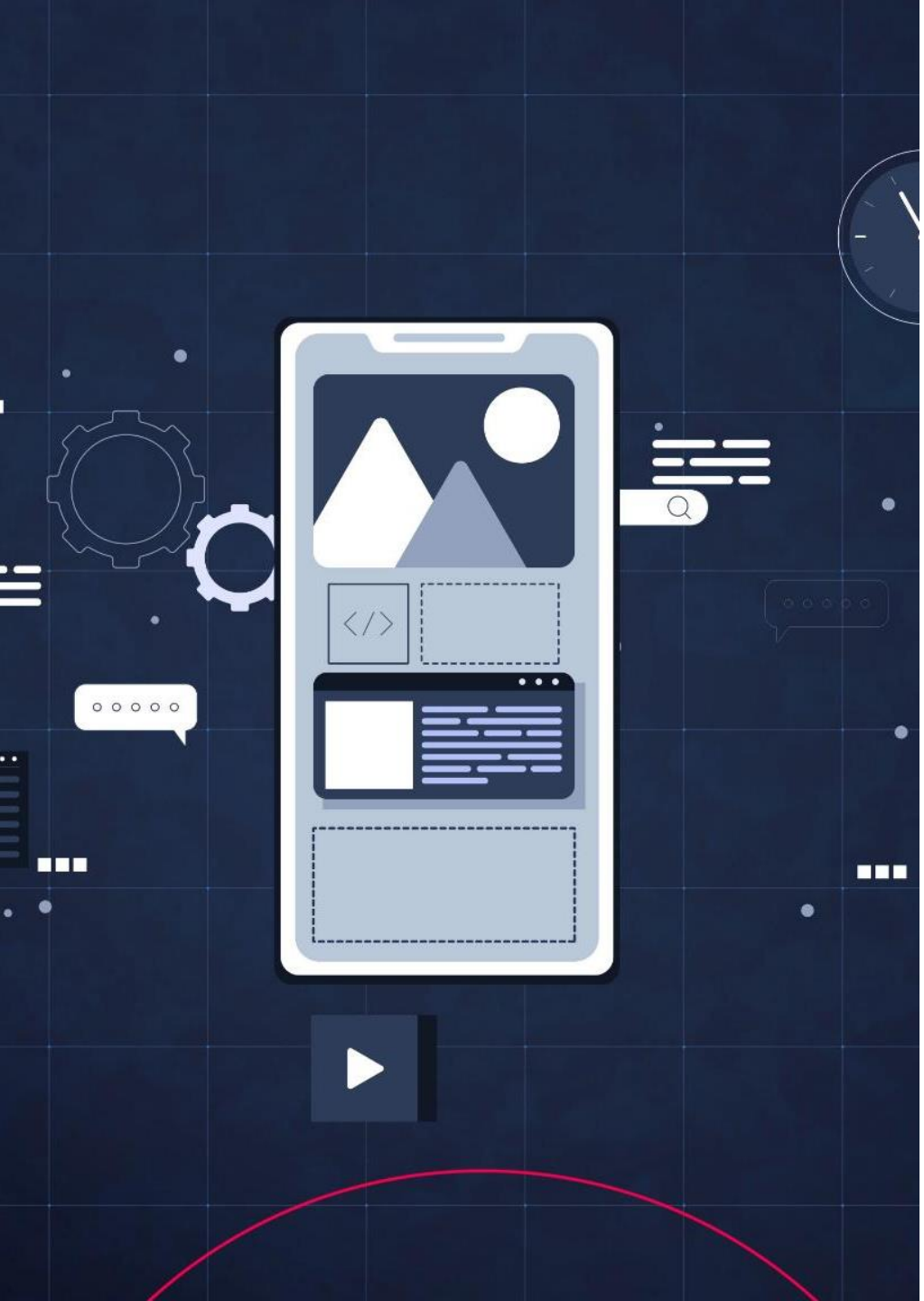
La concurrencia analiza situaciones en las que **dos o más procesos** se ejecuten de forma tal que requieran **acceder a un recurso compartido al mismo tiempo.**

El sistema operativo debe proporcionar mecanismos para mantener la estabilidad del sistema y coordinar las ejecuciones de los procesos involucrados

Multiprogramación vs. Multiprocesamiento

¿Presentan problemas distintos?

Recursos Globales	• El orden de ejecución es crítico
Asignar recursos	• La gestión de recursos es compleja para el SO y puede conducir a interbloqueo
Gestión de errores	• Errores no determinísticos y no reproducibles

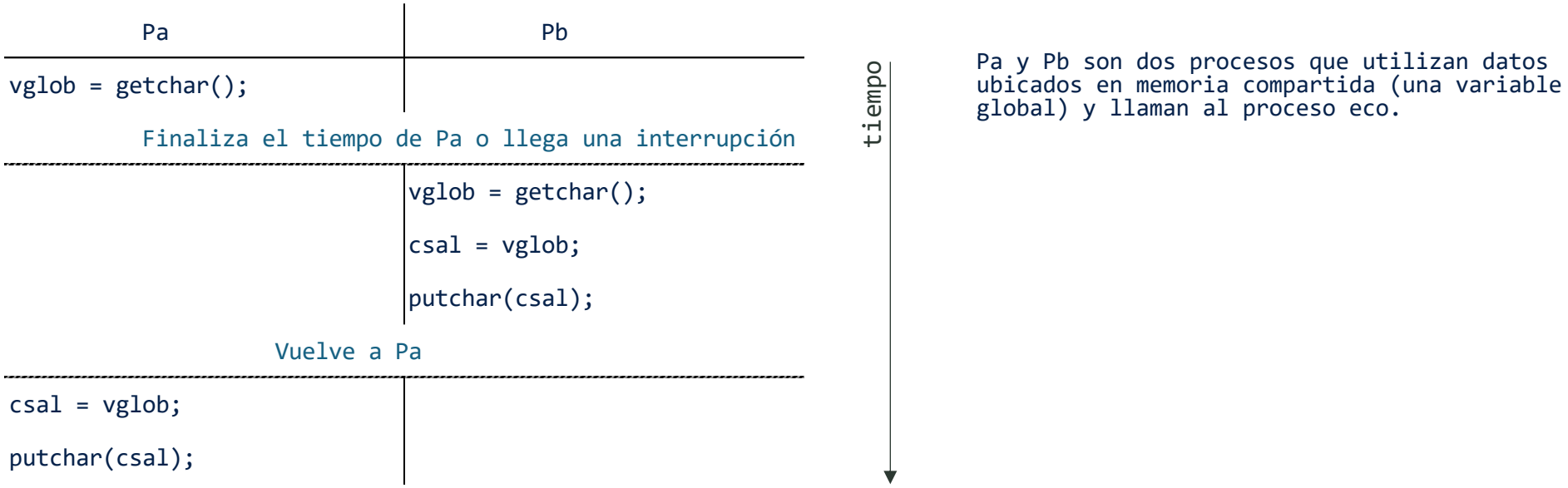


Velocidad relativa de ejecución

No se puede predecir la velocidad relativa de ejecución de un proceso debido a que depende de la actividad de otros procesos, la gestión de las interrupciones y las políticas de planificación del sistema operativo.

Ejemplo

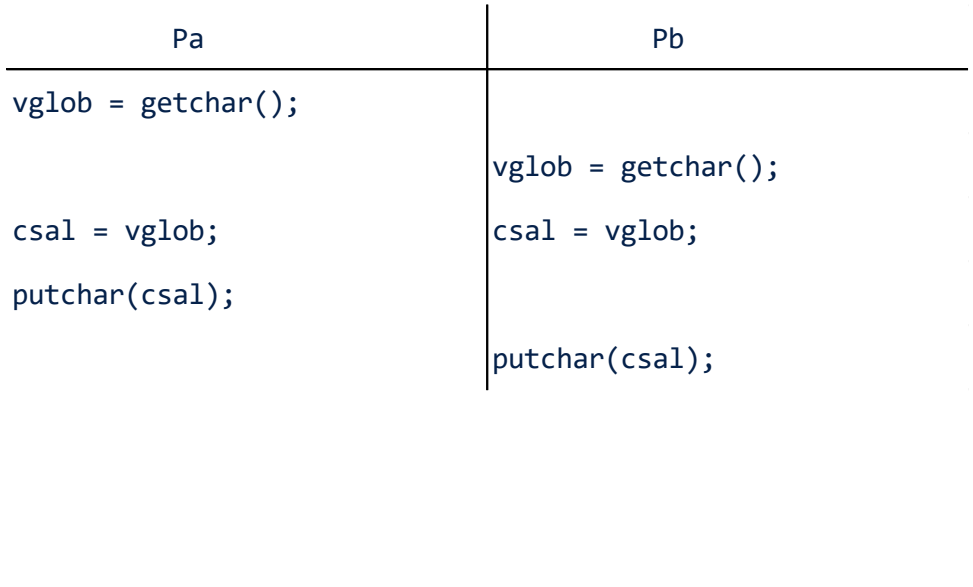
```
// función eco
void eco () {
            vglob = getchar();      //lee un solo carácter desde el teclado y lo asigna a la variable vglob
            csal = vglob;           //La variable csal toma el valor de vglob
            putchar(csal);          //imprime el valor de csal en la pantalla.
}
```



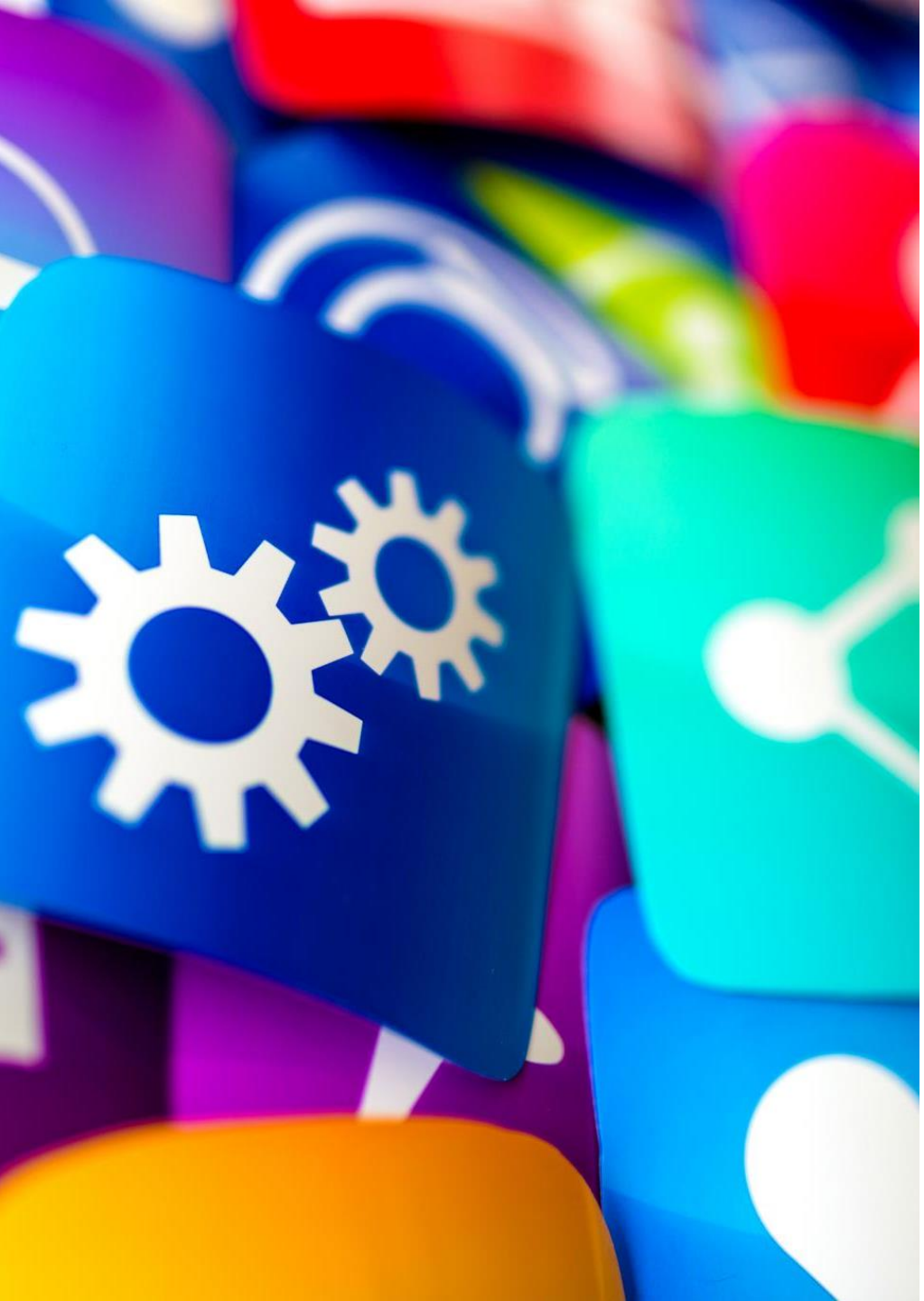
Pa y Pb son dos procesos que utilizan datos ubicados en memoria compartida (una variable global) y llaman al proceso eco.

Mismo ejemplo con dos procesadores

```
// función eco
void eco () {
            vglob = getchar();      //lee un solo carácter desde el teclado y lo asigna a la variable vglob
            csal = vglob;           //La variable csal toma el valor de vglob
            putchar(csal);          //imprime el valor de csal en la pantalla.
}
```



Pa y Pb son dos procesos que utilizan datos ubicados en memoria compartida (una variable global) y llaman al proceso eco.



¿Cómo solucionamos estos problemas?

Hay que proteger los recursos compartidos y la única manera de hacerlo es controlar el código que accede a los mismos.

Condición de carrera

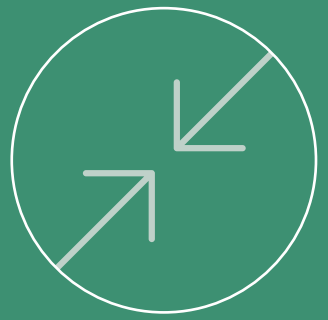
El resultado final de un recurso compartido depende de la coordinación de los procesos, es decir del orden de ejecución.

2. Interacción de procesos

¿Cómo interactúan los procesos?

No se perciben entre si

- Son procesos independientes que no se pretende que trabajen juntos y que no realizan intercambio de información. El SO debe ocuparse de la competencia por los recursos.



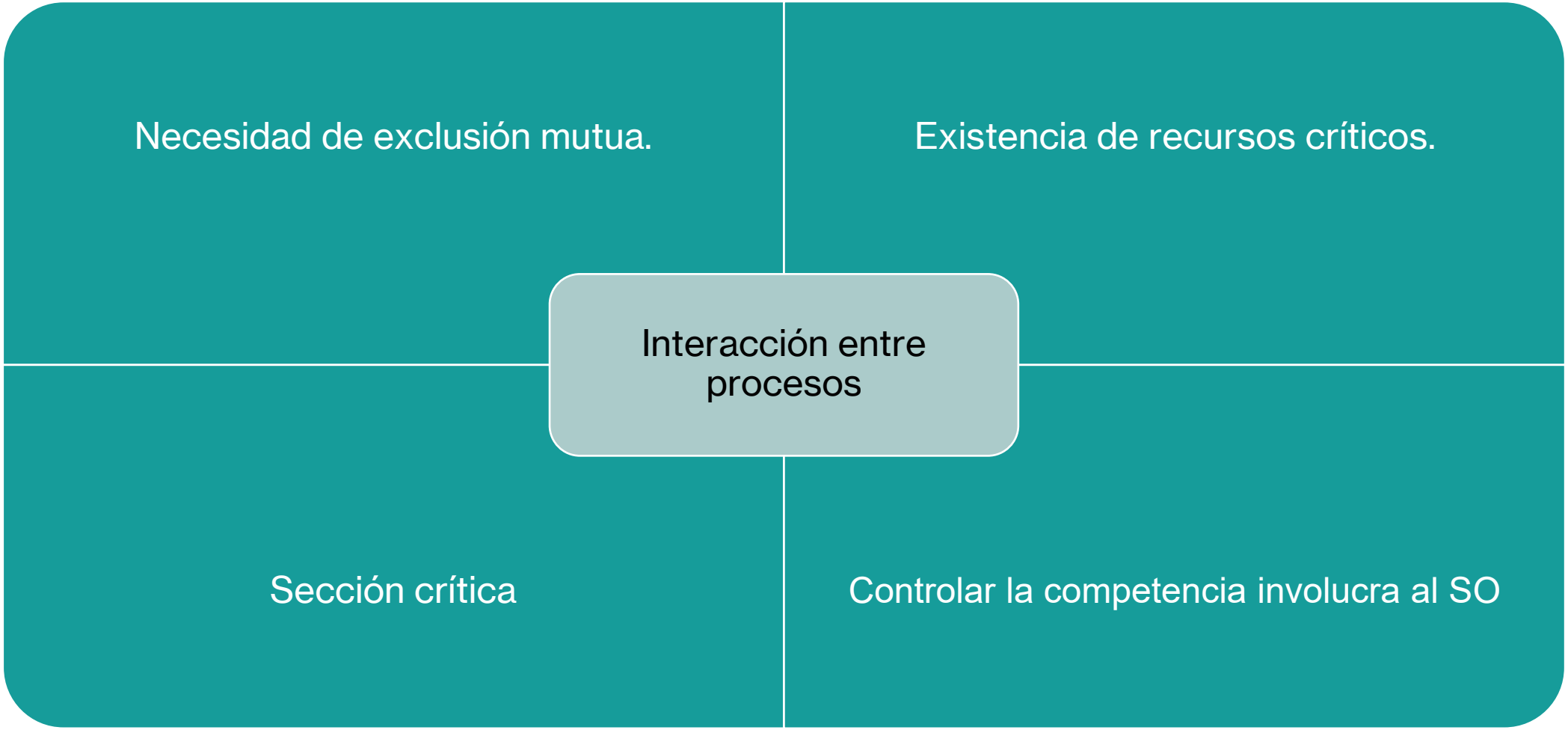
Se perciben indirectamente entre si

- Son procesos, que no tienen conocimiento explícito del resto de los procesos a través de su PID pero saben que comparten el acceso a un objeto común, por ejemplo, un dispositivo de E/S, memoria compartida, etc

Se perciben directamente entre si

- Son procesos que se comunican entre sí ya que conocen el PID del otro y son diseñados para trabajar en conjunto. La comunicación proporciona una manera de sincronizar las actividades que realizan

Conclusiones de la interacción entre procesos



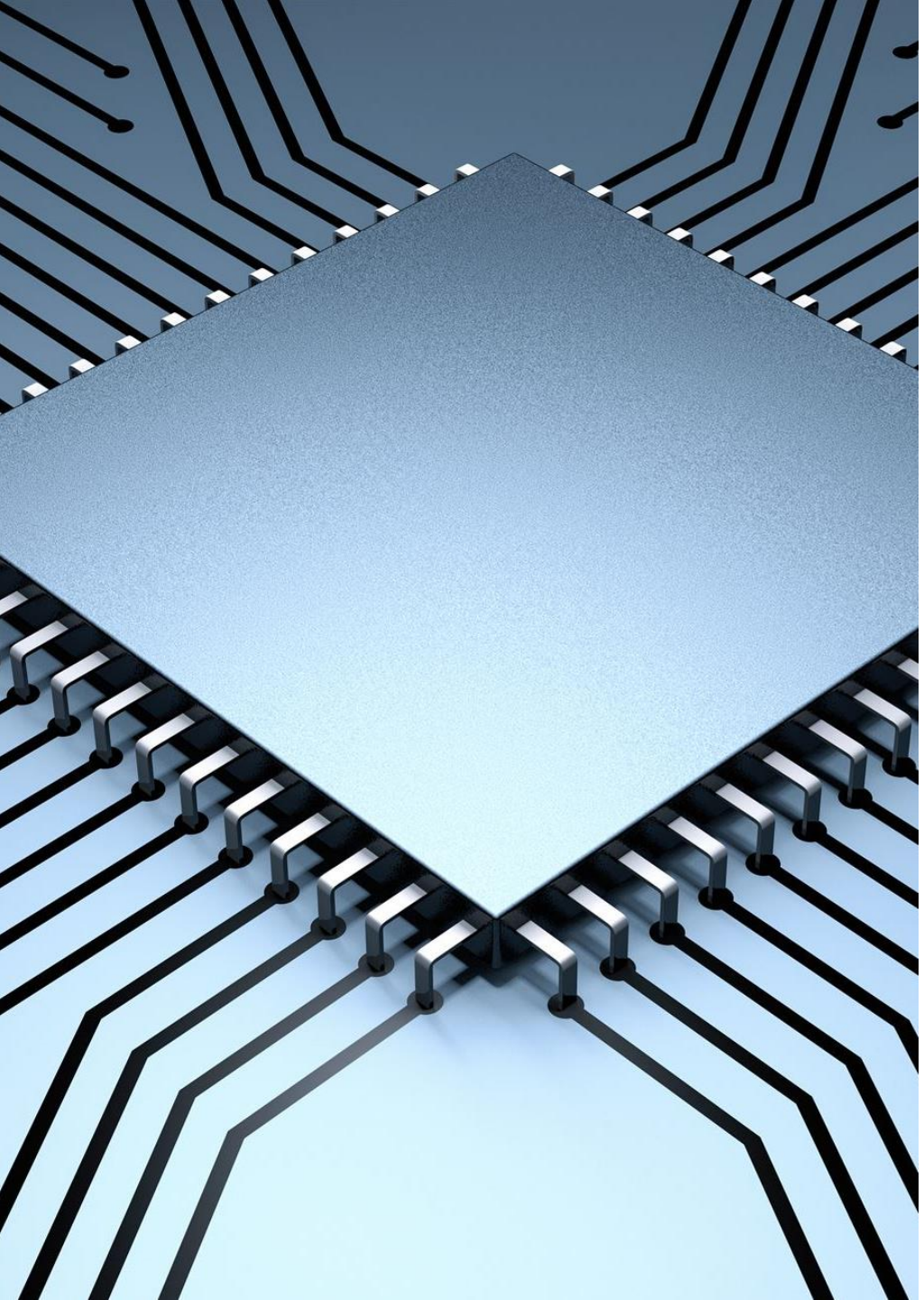
Interacción entre procesos

La exclusión mutua genera...

Deadlock	• Recursos bloqueados hacen que se bloqueen los procesos
Starvation	Procesos que no terminan de ejecutar por falta de recursos

3. Exclusión mutua

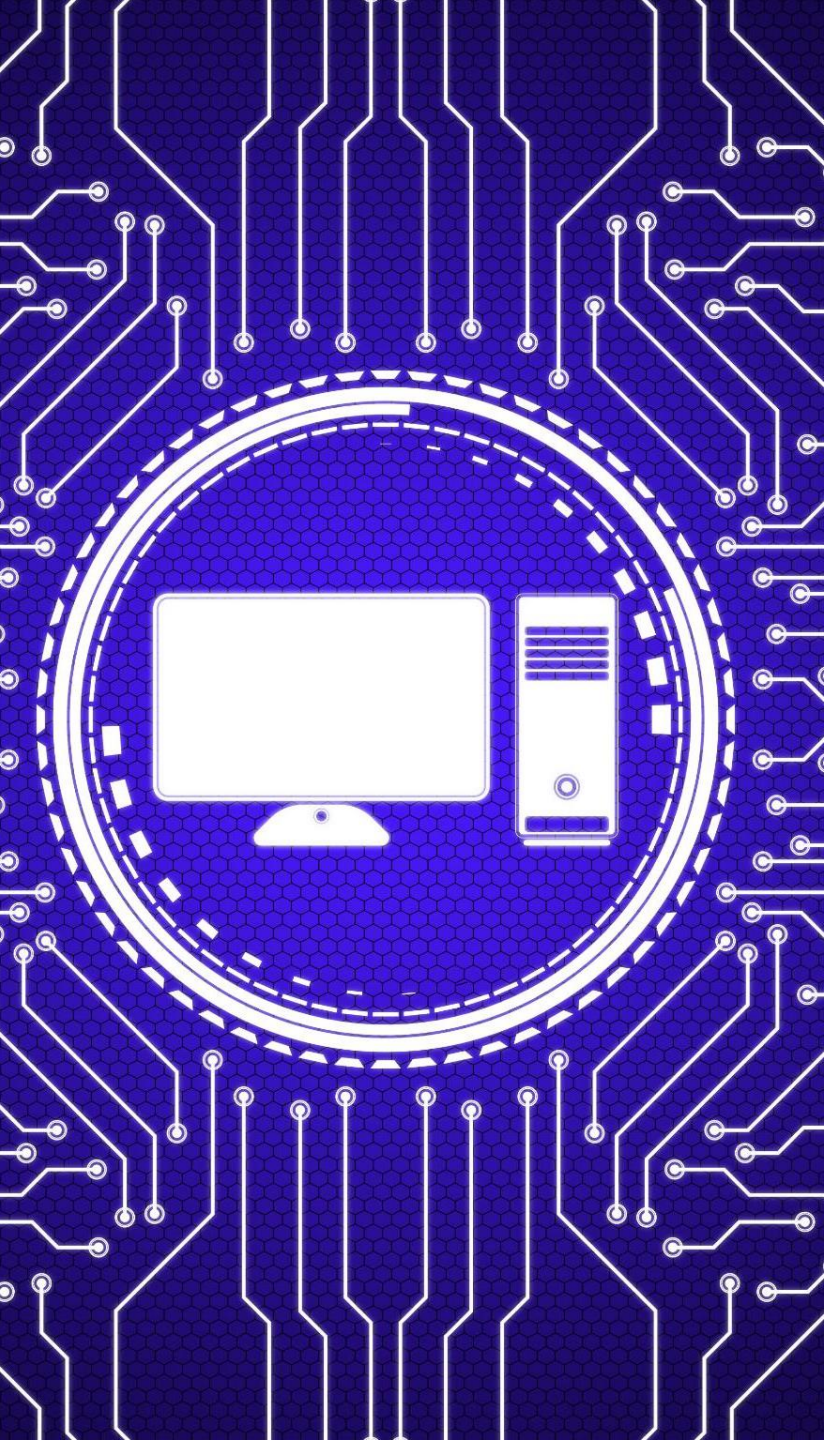
Soporte por Hardware



Deshabilitar interrupciones

Este esquema sólo funciona en máquinas con un único procesador.

Si la sección crítica no puede ser interrumpida, se garantiza la exclusión mutua.



Propiedades del soporte por hardware

Ventajas

- Aplicable a cualquier cantidad de procesos
- Puede utilizarse para dar soporte a varias secciones críticas

Desventajas

- Espera activa
- Posibilidad de deadlock
- Posibilidad de starvation

4. Exclusión mutua

Soporte por Software

Técnicas

Semáforos

- Son estructuras que permiten sincronizar el acceso a un recurso crítico mediante el uso de señales

Monitores

- Es un módulo de software desarrollado en un lenguaje de programación (alto nivel) que proporciona funcionalidad equivalente a un semáforo

Mensajes

- Los mensajes son utilizados por procesos cooperantes para comunicarse y sincronizarse entre ellos

Propiedades de los semáforos

Tipos de semáforos

- “Comunes” o con contador
- Binarios o mutex (valores posibles 0 y 1)

Operaciones

- Un semáforo se define como una variable con un valor entero
- semSignal y semWait para semáforos comunes
- semSignalB y semWaitB para semáforos binarios

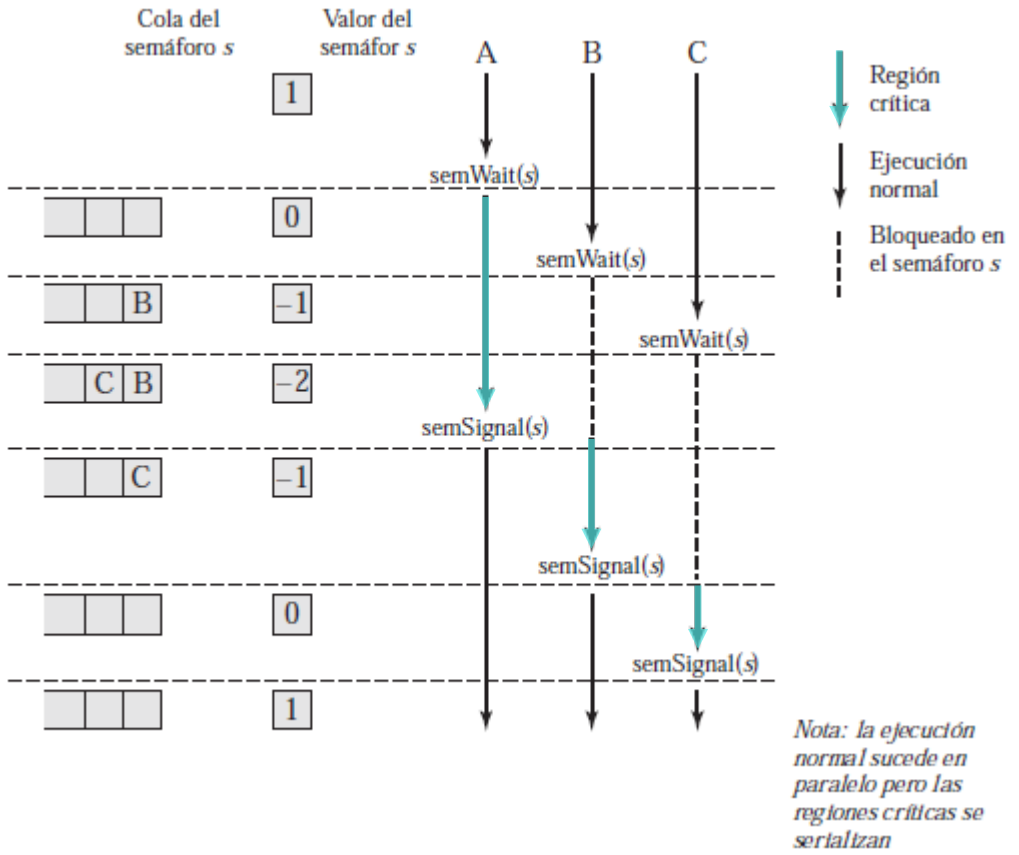
```
struct semaphore {
    int cuenta;
    queueType cola;
}
void semWait(semaphore s)
{
    s.cuenta—;
    if (s.cuenta < 0)
    {
        poner este proceso en s.cola;
        bloquear este proceso;
    }
}
void semSignal(semaphore s)
{
    s.cuenta++;
    if (s.cuenta <= 0)
    {
        extraer un proceso P de s.cola;
        poner el proceso P en la lista de listos;
    }
}
```

- Un semáforo puede ser inicializado a un valor no negativo
- La operación semWait decrementa el valor del semáforo. Si pasa a ser negativo, el proceso que ejecuta semWait se bloquea
- La operación semSignal incrementa el valor del semáforo. Si pasa a ser negativo o cero, se desbloquea uno de los procesos bloqueados con semWait

Ejemplo de uso de semáforos

```
/* programa exclusión mutua */
const int n = /* número de procesos */;
semafore s = 1;
void P(int i)
{
    while (true)
    {
        semWait(s);
        /* sección crítica */;
        semSignal(s);
        /* resto */
    }
}
void main()
{
    paralelos (P(1), P(2), . . . ,P(n));
}
```

Tres procesos A, B y C acceden a un recurso compartido protegido por el semáforo s



UTN.BA FACULTAD REGIONAL BUENOS AIRES

Muchas gracias