**UNIVERSIDAD FRANCISCO DE VITORIA**
**Computación de Alto Rendimiento**

# *COMPUTACIÓN DE ALTO RENDIMIENTO*

---

## Tema 2: Repaso conceptos Programación concurrente

*Grado en Ingeniería Informática*

*Curso 2022/2023*

# Objetivos

- Introducción procesos concurrentes.
- Exclusión mutua.
- Bloqueo mediante el uso de variables compartidas: Peterson, Dekker
- Semáforos
- Sincronización
- Versión más general de semáforos
- Monitores
- Mensajes
- Interbloqueo

## ***Procesos***

Instancia de un programa en ejecución

Unidad mas pequeña de trabajo individualmente planificable por un sistema Operativo. Si estamos hablando de hilos o threads ésta será la Unidad más pequeña planificable.

# Introducción a los procesos concurrentes

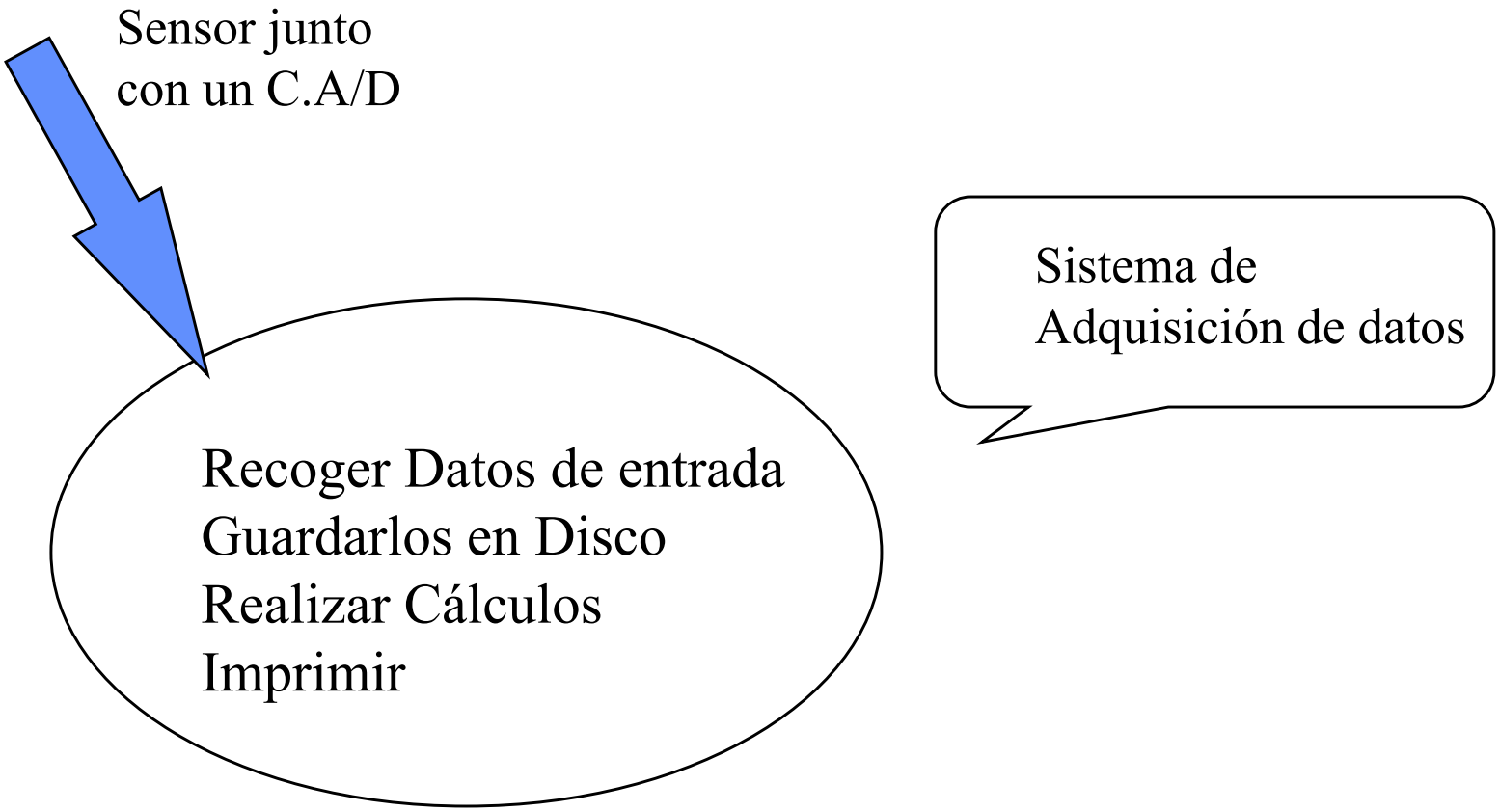
## ***Procesos***

Ventajas de la Operación Multitarea

Ganancia de Velocidad
Uso de dispositivos de E/S que tienen latencia
Conveniencia para el Usuario
Multiprocesamiento (Multiprocesador)
Computación  Distribuida

# Introducción a los procesos concurrentes

## ***Procesos. Ejemplo ejecución secuencial y concurrente***



# Introducción a los procesos concurrentes

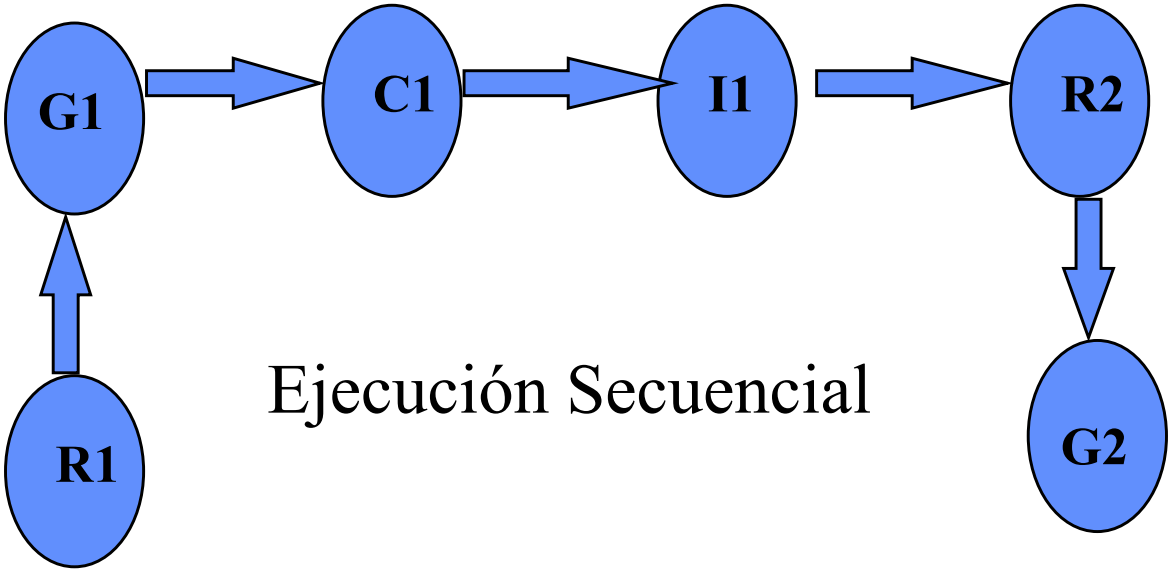
## ***Procesos. Pseudocódigo Ej. Secuencial***

```
Program Adquirir_Sec;
...
Begin
        while TRUE do
          Begin
                RECOGER_AD;
                GUARDAR;
                CALCULAR;
                IMPRIMIR;
        END (While)
END
```

# Introducción a los procesos concurrentes

## ***Procesos. Ej. Secuencial***

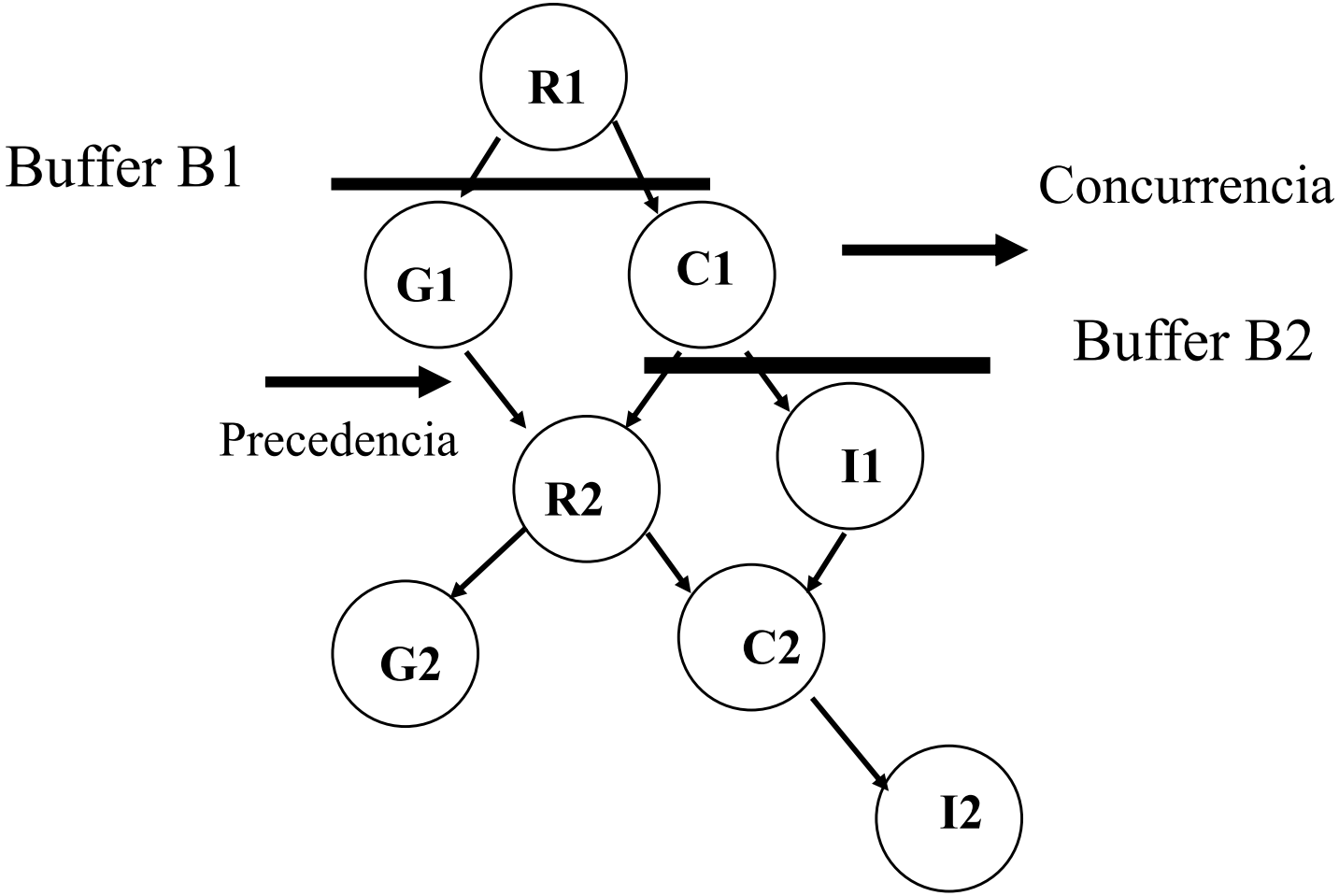
Diagrama de Precedencia



# Introducción a los procesos concurrentes

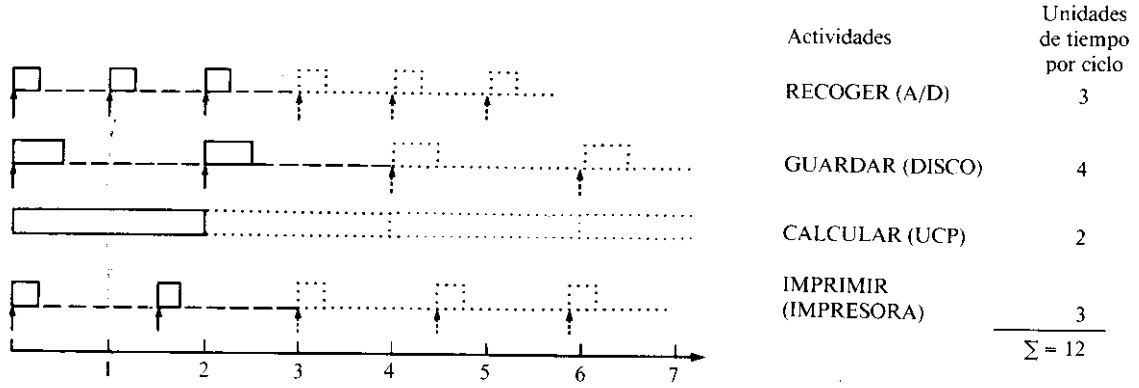
## ***Procesos***

Diagrama de Precedencia. Descomposición y ejecución concurrente para un mismo resultado. Dado que el programa original es cíclico, cada parte tb. lo es.

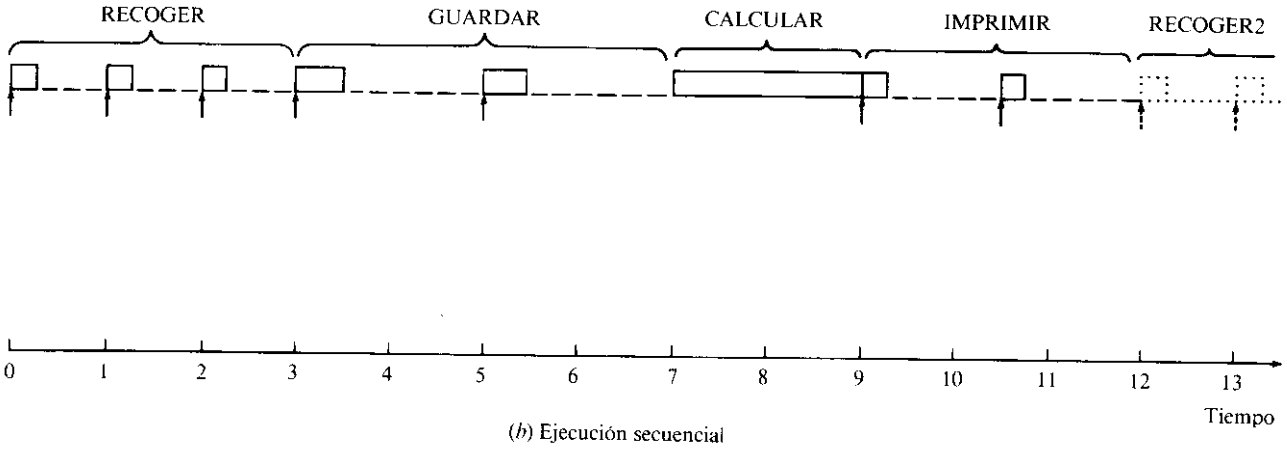


# Introducción a los procesos concurrentes

## *Procesos. Diagrama de ejecución en el tiempo*




(*a*) Suposiciones de tiempo




(*b*) Ejecución secuencial

**Figura 2.2.** Sistema de adquisición de datos.

# Introducción a los procesos concurrentes

## *Procesos. Diagrama de tiempos. Ejecución concurrente Asumiendo prioridades*

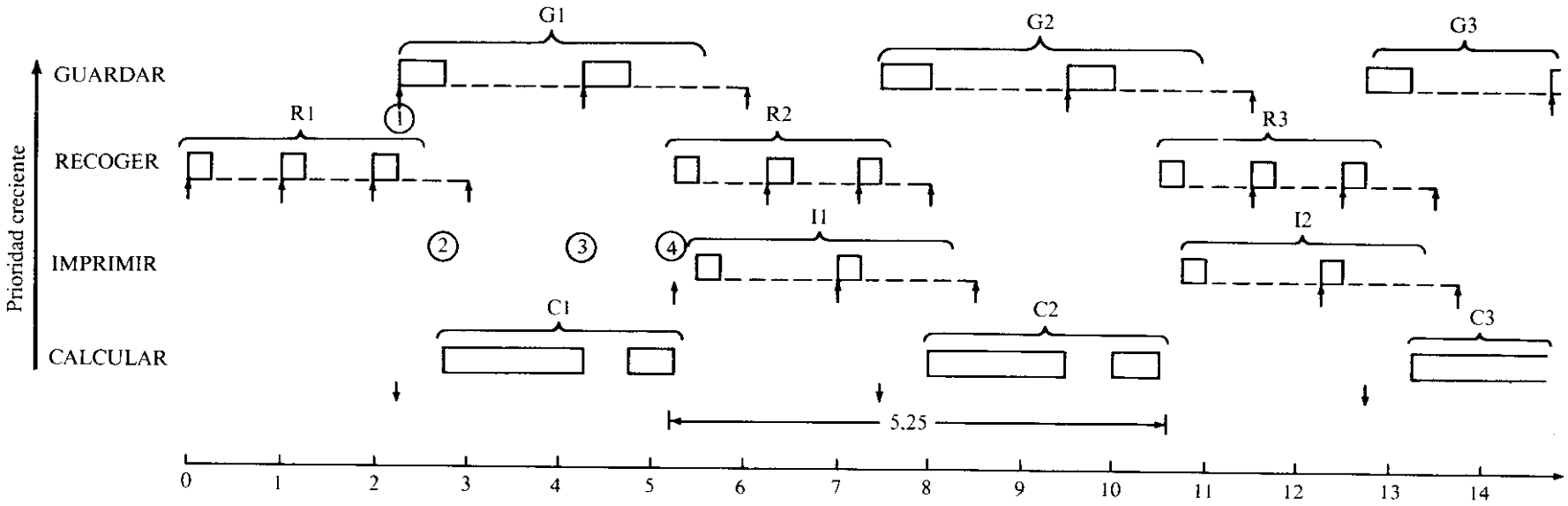



**Figura 2.3.** Diagrama de tiempos de la ejecución del ejemplo multitarea (caso de un solo búfer).

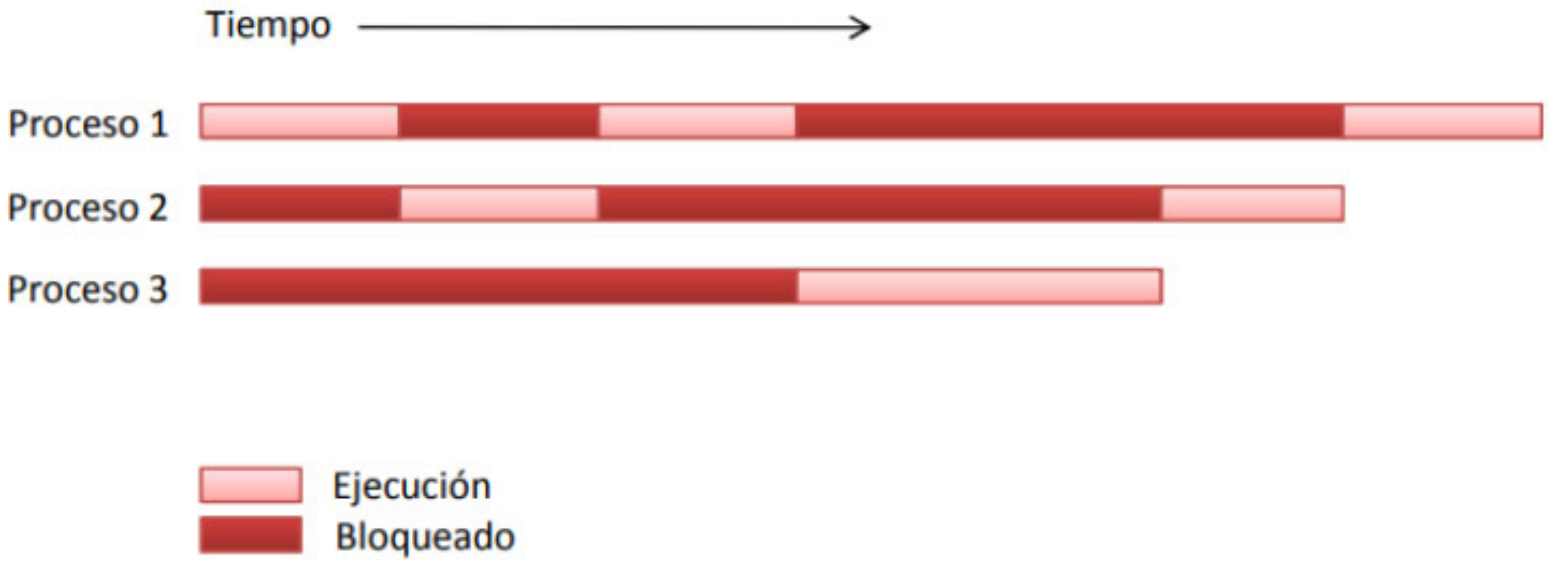
# En general …..

## Procesos concurrentes

- Puntos importantes para el diseño de un Sistema Operativo:
  - **Multiprogramación con un solo procesador: apariencia de ejecución simultánea.**
  - **Multiprocesador: verdadera ejecución simultánea.**
  - **Procesamiento Distribuído: paso de mensajes.**
- La **concurrencia** juega un papel fundamental.

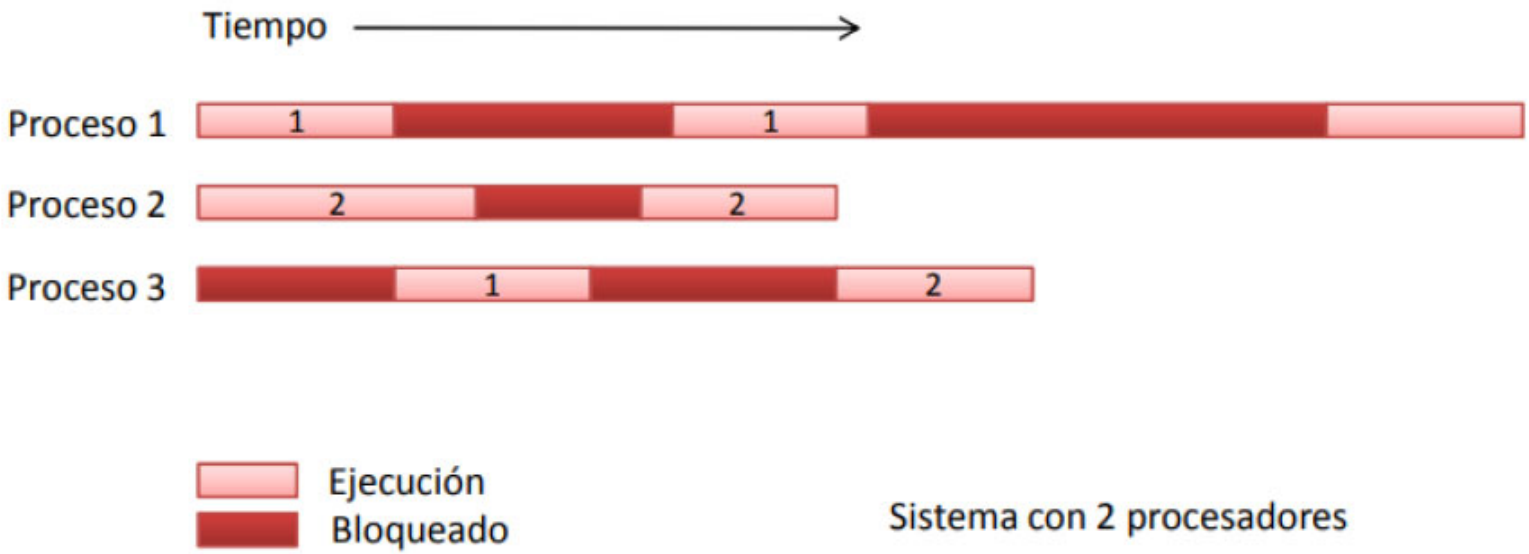
# Intercalado

- **Sistema monoprocesador** donde se **intercala** la ejecución de procesos, lo que aumenta la eficacia.



# Intercalado y solapamiento

- En **sistemas multiprocesador** además del **intercalado** se permite el **solapamiento**.



# Concurrencia

- La concurrencia abarca varios aspectos:
  - **Comunicación entre procesos**
  - **Compartición de recursos**
  - **Sincronización de procesos**
  - **Reserva del procesador** para los procesos

# Interacción entre procesos

- **Procesos Dependientes de forma Indirecta**

  Procesos que tienen algún objeto en común (un buffer de E/S) pero no son conscientes de ello.

  - **Relación:** Competencia por Compartición.
  - **Influencia sobre otros:** sus acciones pueden afectar a los resultados de otros procesos.
  - **Problemas:** Exclusión mutua, Interbloqueo, Inanición, Coherencia de datos.

# Competencia entre procesos por Compartición

- Procesos que interaccionan con otros procesos sin tener conocimiento explícito de ellos.
- **Comparten variables, archivos, bases de datos, etc.**
- Mismos problemas que con procesos independientes porque los datos son recursos.
- Se añade el problema de la **coherencia de datos**.

# Interacción entre procesos

- **Procesos Dependientes de forma Directa**

  Procesos que se comunican con otros procesos y que pueden ser diseñados para trabajar conjuntamente.

  - **Relación:** Cooperación por Comunicación.
  - **Influencia sobre otros:** sus resultados pueden depender de la información obtenida de otros.
  - **Problemas:** Interbloqueo, Inanición.

# Competencia entre procesos por Comunicación

- En este caso los procesos conocen las existencia de los demás y cooperan y comparten recursos.
- Dado que se comunican, se pueden **sincronizar** y **coordinar.**
- Siguen apareciendo problemas de **interbloqueo** (por ejemplo por la propia comunicación) e **inanición**.

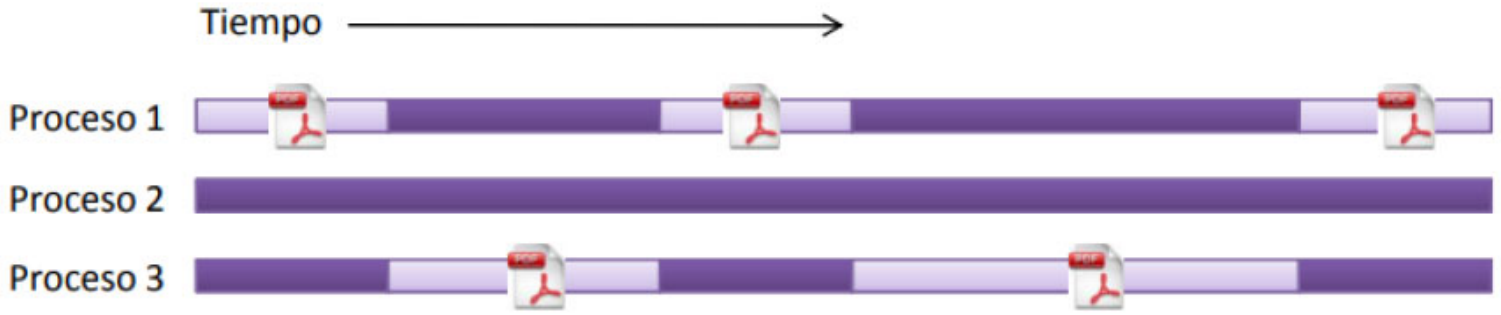
# Problemas asociados a la concurrencia

- Los procesos (hilos) concurrentes suelen compartir datos (ficheros, memoria común) y recursos.
- Si el acceso a estos datos o recursos no se controla se puede tener inconsistencia.
- La velocidad de ejecución de los procesos no se puede predecir.Con lo que pueden existir problemas al compartir datos.

# Problemas asociados a la concurrencia

## Competencia entre procesos y recursos

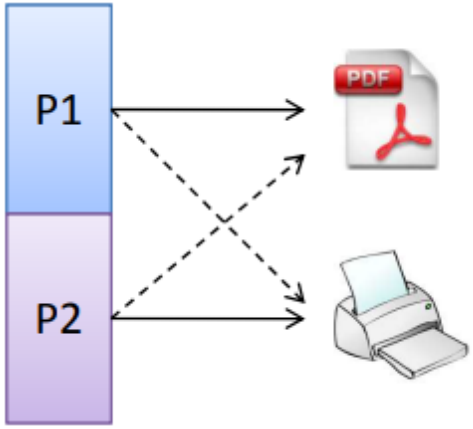
- **Inanición**:
  - Tres procesos necesitan un recurso.
  - El S.O. da acceso alternativamente a P1 y P3
  - P2 queda bloqueado indefinidamente



# Problemas asociados a la concurrencia

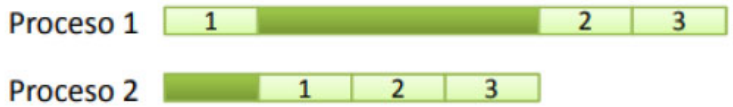
## Competencia entre procesos y recursos

- **Interbloqueo**:
  - Los dos procesos necesitan dos recursos y el sistema le asigna uno a cada uno.
  - No liberan el que tienen asignado hasta que se libere el otro



- Sistema monoprocesador
- Método echo() en memoria al que pueden acceder varios procesos al mismo tiempo.
  - *Se ahorra memoria.*
  - *Pero acarrea problemas si varios procesos acceden al mismo tiempo.*

```
void echo()
{
        1 chin = getchar();
        2 chout = chin;
        3 putchar(chout);
}
```

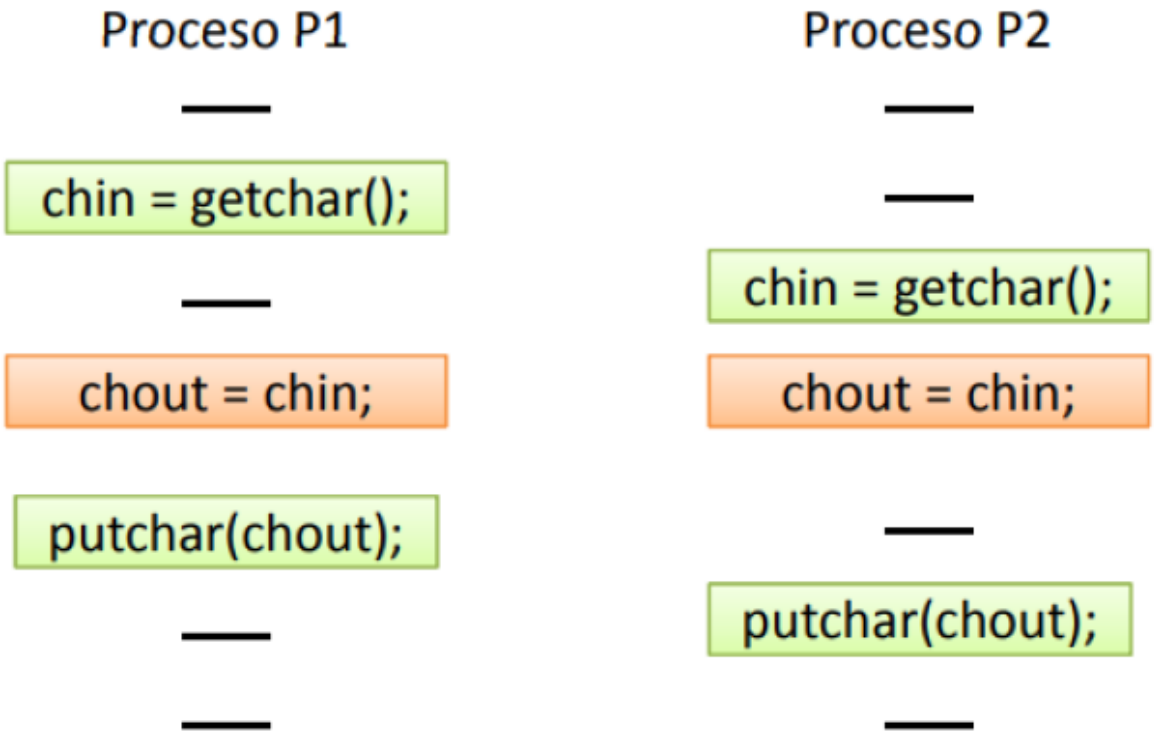


- Problema:
  - En el ejemplo se pierde el carácter del proceso 1 y se repite dos veces el del proceso 2.

- Solución: definir sección crítica
  - La solución radica en bloquear el proceso 2 hasta que el proceso 1 termine.

  - Al terminar el proceso 1, el proceso 2 se desbloquea y continúa su ejecución.

- Sistema multiprocesador.
  - Dos procesos llaman al método echo().
  - Se produce el mismo error.
  - Solución: igual a la anterior



- Calcula la suma de los N primeros números utilizando procesos ligeros.

```
int suma_total = 0;

void suma_parcial(int ni, int nf)
 {
    int j = 0;
    int suma parcial = 0;

    for (j = ni; j <= nf; j++)
      suma_parcial = suma_parcial +j;

    suma_total = suma_total +
    suma_parcial;
    pthread_exit(0);
 }
```

- Si varios procesos ejecutan concurrentemente este código se puede obtener un resultado incorrecto.

- Solución: **secciones críticas**

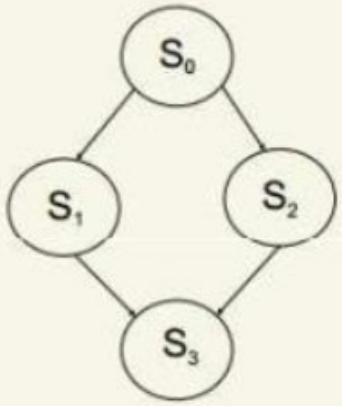
```
void suma_parcial(int ni, int nf)
{
        int j = 0;
        int suma_parcial = 0;

        for (j = ni; j <= nf; j++)
          suma_parcial = suma_parcial + j;

        Entrada en la sección crítica
         suma_total = suma_total+
         suma_parcial;
        Salida de la sección crítica

        pthread_exit(0);
}
```

# Problemas asociados a la concurrencia

## Secciónes Críticas ---->Indeterminismo-Soluciones no coherentes

Al compartir datos entre procesos se pueden producir problemas de indeterminismo (resultados diferentes según escenario de prueba).

Ejemplo: $S_1$ y $S_2$ no son independientes, sino que comparten la variable $x$.




$S_0$: $x = 100$;
$S_1$: $x := x + 10$;
$S_2$: if $x > 100$ then write($x$);
else write ($x - 50$);
$S_3$: nop;

*Escenario #1*: $S_1$ y $S_2 \Rightarrow$ Se escribe $x = 110$.

*Escenario #2*: $S_2$ y $S_1 \Rightarrow$ Se escribe $x = 50$.

*Escenario #3*: $S_2$ pierde el control (p. ej. fin de quantum) antes de escribir y sigue $S_1 \Rightarrow$ Se escribe $x = 60$.