

Capítulo 5

Concurrencia: exclusión mutua y sincronización


A decorative vertical bar on the left side of the slide, featuring a light red background with faint, overlapping technical diagrams. These diagrams include a network topology at the top, a sequence of process flowcharts in the middle, and a database cylinder icon at the bottom.

Concurrencia

- Comunicación entre procesos.
- Compartición y competencia por los recursos.
- Sincronización de la ejecución de varios procesos.
- Asignación del tiempo de procesador a los procesos.

Concurrencia

- Múltiples aplicaciones:
 - Multiprogramación.
- Aplicaciones estructuradas:
 - Algunas aplicaciones pueden implementarse eficazmente como un conjunto de procesos concurrentes.
- Estructura del sistema operativo:
 - Algunos sistemas operativos están implementados como un conjunto de procesos o hilos.



Dificultades con la concurrencia

- Compartir recursos globales.
- Gestionar la asignación óptima de recursos.
- Localizar un error de programación.

Un ejemplo sencillo

```
void echo()  
{  
    ent = getchar();  
    sal = ent;  
    putchar(sal);  
}
```

Un ejemplo sencillo

Proceso P1

```
.  
ent = getchar() ;  
.   
sal = ent ;  
putchar(sal) ;  
.   
.
```

Proceso P2

```
.  
.   
ent = getchar() ;  
sal = ent ;  
.   
putchar(sal) ;  
.
```

Labores del sistema operativo

- Seguir la pista de los distintos procesos activos.
- Asignar y retirar los recursos:
 - Tiempo de procesador.
 - Memoria.
 - Archivos.
 - Dispositivos de E/S.
- Proteger los datos y los recursos físicos.
- Los resultados de un proceso deben ser independientes de la velocidad relativa a la que se realiza la ejecución de otros procesos concurrentes.




Interacción entre procesos

- Los procesos no tienen conocimiento de los demás.
- Los procesos tienen un conocimiento indirecto de los otros.
- Los procesos tienen un conocimiento directo de los otros.

Competencia entre procesos por los recursos

- Exclusión mutua
 - Secciones críticas:
 - Sólo un programa puede acceder a su sección crítica en un momento dado.
 - Por ejemplo, sólo se permite que un proceso envíe una orden a la impresora en un momento dado.
- Interbloqueo.
- Inanición.



Cooperación entre procesos por compartimiento

- Las operaciones de escritura deben ser mutuamente excluyentes.
- Las secciones críticas garantizan la integridad de los datos.

Cooperación entre procesos por comunicación

- Paso de mensajes:
 - No es necesario el control de la exclusión mutua.
- Puede producirse un interbloqueo:
 - Cada proceso puede estar esperando una comunicación del otro.
- Puede producirse inanición:
 - Dos procesos se están mandando mensajes mientras que otro proceso está esperando recibir un mensaje.

Requisitos para la exclusión mutua

- Sólo un proceso debe tener permiso para entrar en la sección crítica por un recurso en un instante dado.
- Un proceso que se interrumpe en una sección crítica debe hacerlo sin interferir con los otros procesos.
- No puede permitirse el interbloqueo o la inanición.

Requisitos para la exclusión mutua

- Cuando ningún proceso está en su sección crítica, cualquier proceso que solicite entrar en la suya debe poder hacerlo sin dilación.
- No se deben hacer suposiciones sobre la velocidad relativa de los procesos o el número de procesadores.
- Un proceso permanece en su sección crítica sólo por un tiempo finito.

Primer intento

- Espera activa:
 - Un proceso siempre está esperando a entrar en su sección crítica.
 - Un proceso no puede hacer nada productivo hasta que obtiene permiso para entrar en su sección crítica.

Corrutinas

- Diseñadas para poder pasar el control de la ejecución entre ellas.
- No es una técnica apropiada para dar soporte al procesamiento concurrente.

Segundo intento

- Cada proceso puede examinar el estado del otro pero no lo puede alterar.
- Cuando un proceso desea entrar en su sección crítica comprueba en primer lugar el otro proceso.
- Si no hay otro proceso en su sección crítica fija su estado para la sección crítica.
- Este método no garantiza la exclusión mutua.
- Cada proceso puede comprobar las señales y luego entra en la sección crítica al mismo tiempo.

Tercer intento

- Dar valor a la señal para entrar en la sección crítica antes de comprobar otros procesos.
- Si hay otro proceso en la sección crítica cuando se ha dado valor a la señal, el proceso queda bloqueado hasta que el otro proceso abandona la sección crítica.
- Se provocará un interbloqueo cuando dos procesos den valor a sus señales antes de entrar en la sección crítica. Entonces cada proceso debe esperar a que el otro abandone la sección crítica.

Cuarto intento

- Un proceso activa su señal para indicar que desea entrar en la sección crítica, pero debe estar listo para desactivar la variable señal.
- Se comprueban los otros procesos. Si están en la sección crítica, la señal se desactiva y luego se vuelve a activar para indicar que desea entrar en la sección crítica. Esta operación se repite hasta que el proceso puede entrar en la sección crítica.

Cuarto intento

- Es posible que cada proceso active su señal, compruebe otros procesos y desactive su señal. Esta situación no se mantendrá por mucho tiempo, de modo que no se trata de un interbloqueo. Así pues, se rechaza el cuarto intento.



Una solución correcta

- Cada proceso tiene un turno para entrar en la sección crítica.
- Si un proceso desea entrar en la sección crítica, debe activar su señal y puede que tenga que esperar a que llegue su turno.

Exclusión mutua: soluciones por hardware

- Inhabilitación de interrupciones:
 - Un proceso continuará ejecutándose hasta que solicite un servicio del sistema operativo o hasta que sea interrumpido.
 - Para garantizar la exclusión mutua es suficiente con impedir que un proceso sea interrumpido.
 - Se limita la capacidad del procesador para intercalar programas.
 - Multiprocesador:
 - Inhabilitar las interrupciones de un procesador no garantiza la exclusión mutua.

Exclusión mutua: soluciones por hardware

- Instrucciones especiales de máquina:
 - Se realizan en un único ciclo de instrucción.
 - No están sujetas a injerencias por parte de otras instrucciones.
 - Leer y escribir.
 - Leer y examinar.

Exclusión mutua: soluciones por hardware


- La instrucción Comparar y Fijar

```
booleano TS (int i)  
{  
    if (i == 0) {  
        i = 1;  
        return cierto;  
    }  
    else {  
        return falso;  
    }  
}
```

Exclusión mutua: soluciones por hardware

- La instrucción Intercambiar

```
void intercambiar(int registro,
                  int memoria)
{
    int temp;
    temp = memoria;
    memoria = registro;
    registro = temp;
}
```

Instrucciones de máquina y exclusión mutua

- Ventajas:
 - Es aplicable a cualquier número de procesos en sistemas con memoria compartida, tanto de monoprocesador como de multiprocesador.
 - Es simple y fácil de verificar.
 - Puede usarse para disponer de varias secciones críticas.



Instrucciones de máquina y exclusión mutua


- Desventajas:
 - La espera activa consume tiempo del procesador.
 - Puede producirse inanición cuando un proceso abandona la sección crítica y hay más de un proceso esperando.
 - Interbloqueo:
 - Si un proceso con baja prioridad entra en su sección crítica y existe otro proceso con mayor prioridad, entonces el proceso cuya prioridad es mayor obtendrá el procesador para esperar a poder entrar en la sección crítica.

Semáforos

- Para la señalización se utiliza una variable especial llamada semáforo.
- Si un proceso está esperando una señal, el proceso es suspendido hasta que tenga lugar la transmisión de la señal.
- Las operaciones *wait* y *signal* no pueden ser interrumpidas.
- Se emplea una cola para mantener los procesos esperando en el semáforo.

Semáforos

- Un semáforo es una variable que tiene un valor entero:
 - Puede iniciarse con un valor no negativo.
 - La operación *wait* disminuye el valor del semáforo.
 - La operación *signal* incrementa el valor del semáforo.



Problema del productor/consumidor

- Uno o más productores generan datos y los sitúan en un buffer.
- Un único consumidor saca elementos del buffer de uno en uno.
- Sólo un productor o consumidor puede acceder al buffer en un instante dado.

Producer

```
producer:  
while (cierto) {  
    /* producir elemento v */  
    b[ent] = v;  
    ent++;  
}
```

Consumidor

```
consumidor:
while (cierto) {
    while (ent <= sal)
        /*no hacer nada */;
    w = b[sal];
    sal++;
    /* consumir elemento w */
}
```

Productor con buffer circular

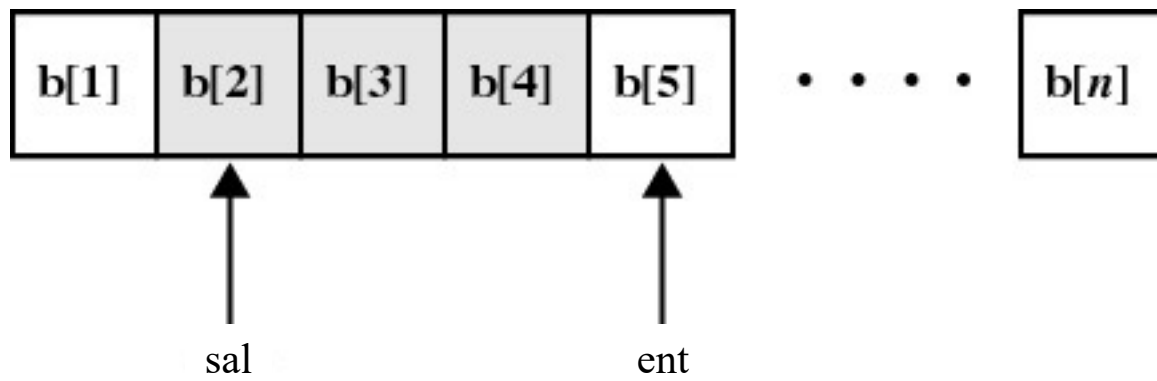
productor:

```
while (cierto) {  
    /* producir elemento v */  
    while ((ent + 1) % n == sal)  
        /* no hacer nada */;  
    b[ent] = v;  
    ent = (ent + 1) % n  
}
```

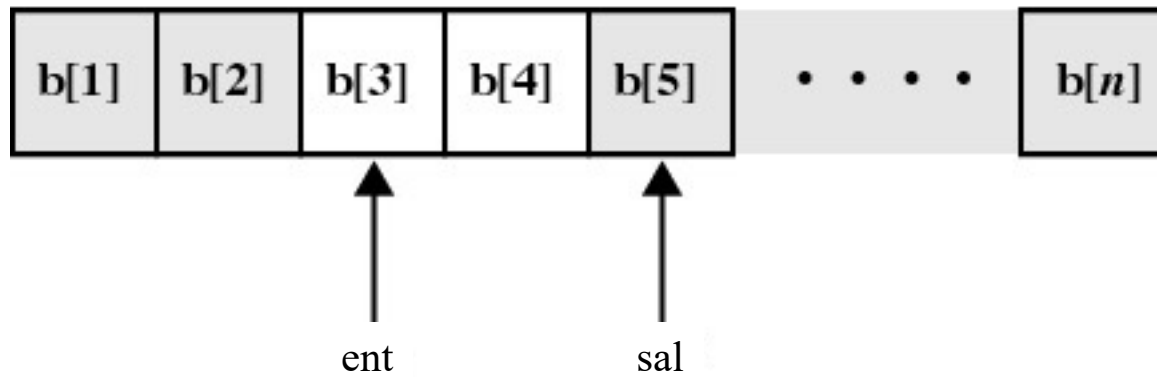

Consumidor con buffer circular

consumidor:

```
while (cierto) {  
    while (ent == sal)  
        /* no hacer nada */;  
    w = b[sal];  
    sal = (sal + 1) % n;  
    /* consumir elemento w */  
}
```



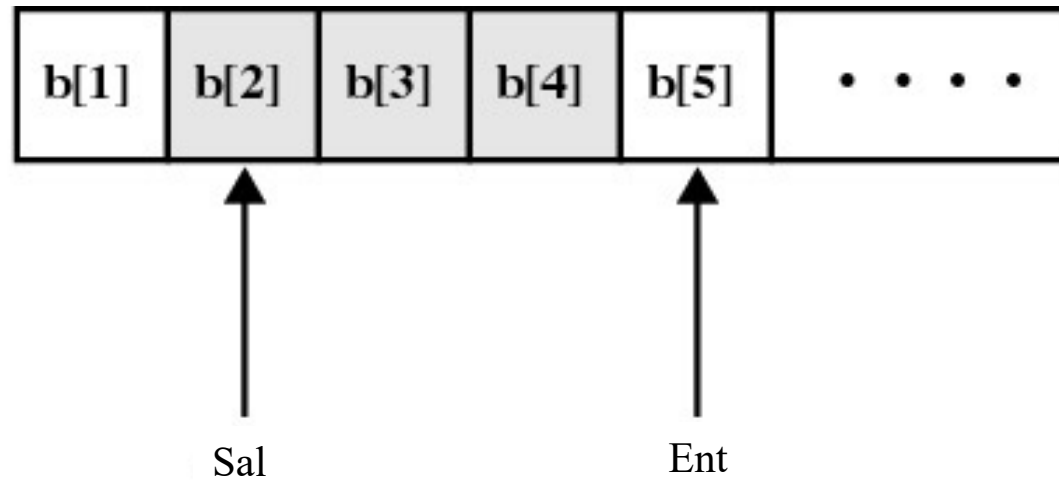
(a)



(b)

Figura 5.15. Buffer circular limitado en el problema del productor/consumidor.

Buffer ilimitado



Nota: El área sombreada indica la parte del buffer ocupada.

Figura 5.11. Buffer ilimitado en el problema del productor/consumidor.

Problema de la barbería

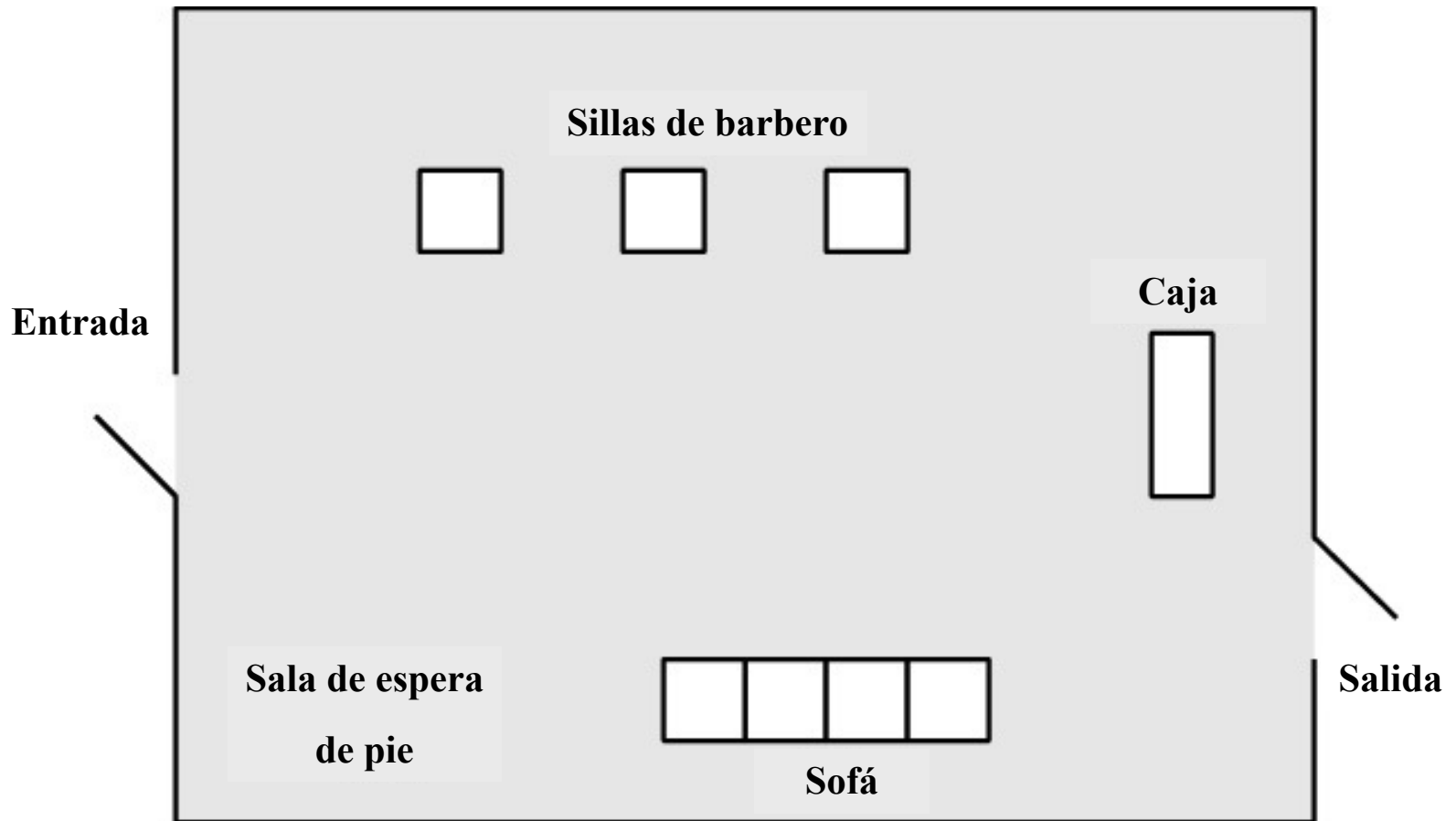


Figura 5.18. La barbería.

Monitores

- Un monitor es un módulo de software.
- Características básicas:
 - Las variables de datos locales están sólo accesibles para el monitor.
 - Un proceso entra en el monitor invocando a uno de sus procedimientos.
 - Sólo un proceso se puede estar ejecutando en el monitor en un instante dado.

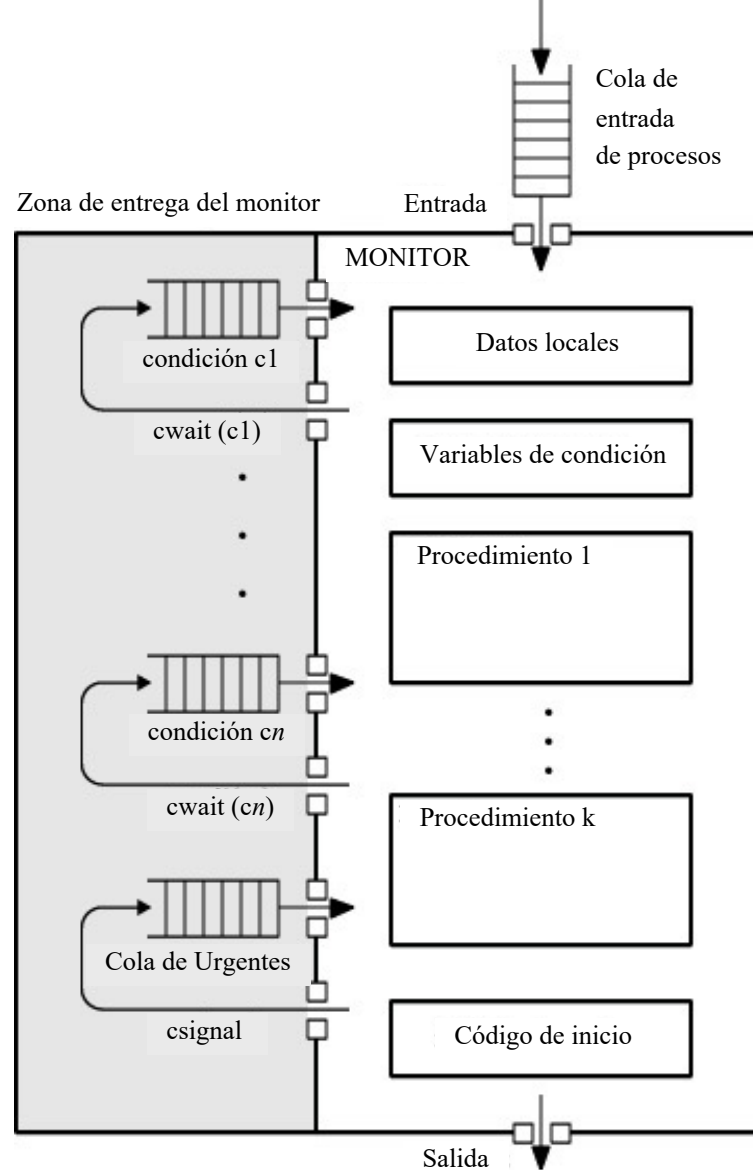


Figura 5.21. Estructura de un monitor.

Paso de mensajes

- Refuerzo de la exclusión mutua.
- Intercambio de información.

send (destino, mensaje)

receive (origen, mensaje)

Sincronización

- El emisor y el receptor pueden ser bloqueantes o no bloqueantes (esperando un mensaje).
- Envío bloqueante, recepción bloqueante:
 - Tanto el emisor como el receptor se bloquean hasta que se entrega el mensaje.
 - Esta técnica se conoce como *rendezvous*.

Sincronización

- Envío no bloqueante, recepción bloqueante:
 - Permite que un proceso envíe uno o más mensajes a varios destinos tan rápido como sera posible.
 - El receptor se bloquea hasta que llega el mensaje solicitado.
- Envío no bloqueante, recepción no bloqueante:
 - Nadie debe esperar.

Direcccionamiento

- Direcccionamiento directo:
 - La primitiva *send* incluye una identificación específica del proceso de destino.
 - La primitiva *receive* puede conocer de antemano de qué proceso espera un mensaje.
 - La primitiva *receive* puede utilizar el parámetro *origen* para devolver un valor cuando se haya realizado la operación de recepción.

Direccionamiento

- Direccionamiento indirecto:
 - Los mensajes se envían a una estructura de datos compartida formada por colas.
 - Estas colas se denominan buzones (*mailboxes*).
 - Un proceso envía mensajes al buzón apropiado y el otro los coge del buzón.

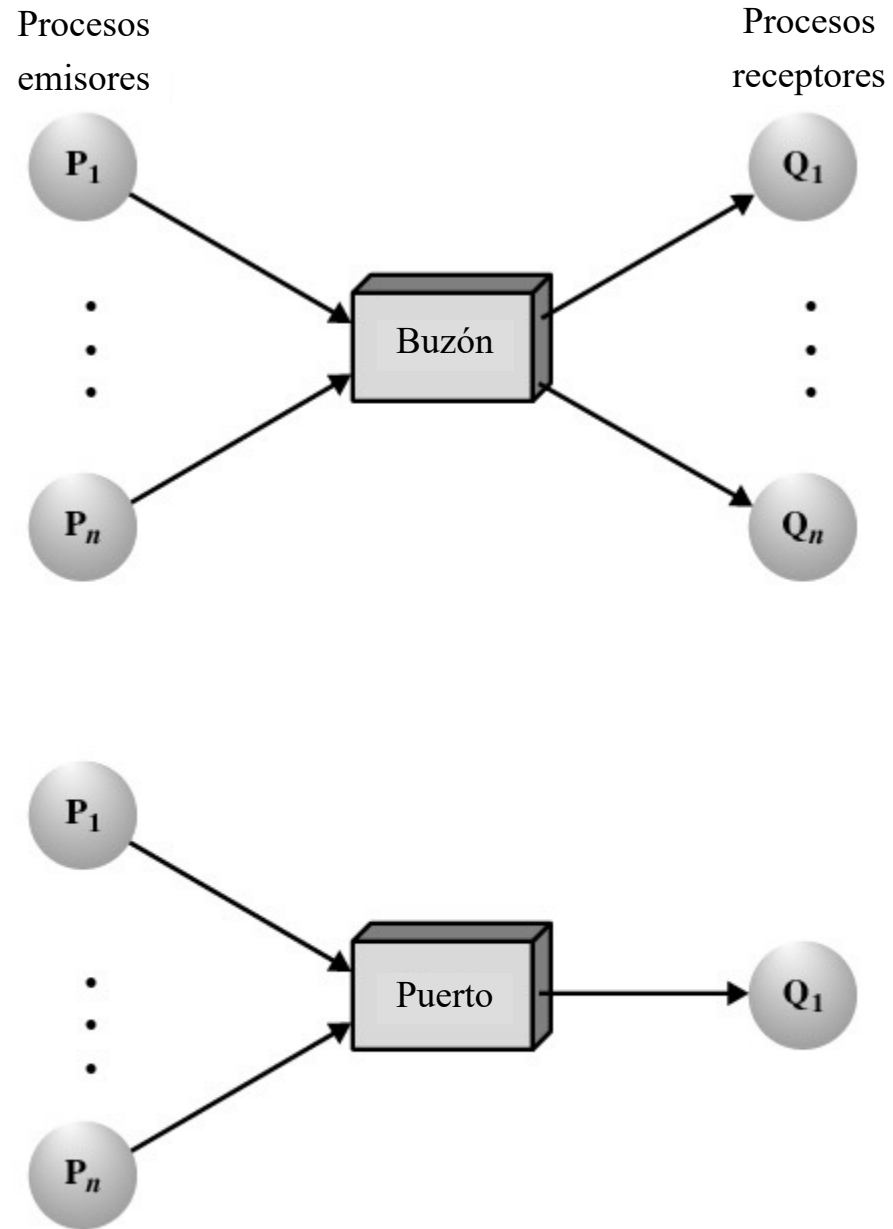


Figura 5.24. Comunicación indirecta entre procesos.

Formato de mensajes

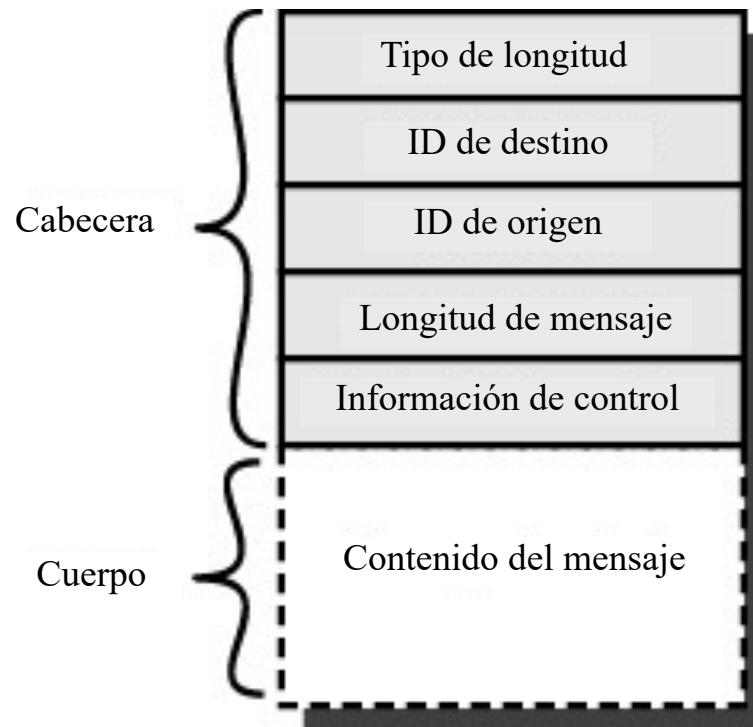



Figura 5.25. Formato típico de mensaje.



Problema de los lectores/escritores

- Cualquier número de lectores puede leer el archivo simultáneamente.
- Sólo puede escribir en el archivo un escritor en cada instante.
- Si un escritor está accediendo al archivo, ningún lector puede leerlo.