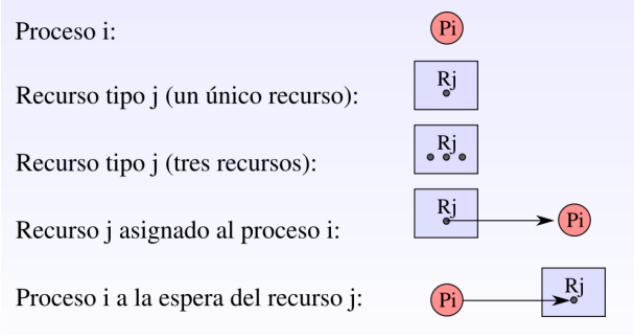
Interbloqueo e inanición

El interbloqueo se define como el **bloqueo permanente** de un conjunto de procesos que compiten por los recursos o bien se comunican unos con otros. Existen necesidades conflictivas por los recursos por parte de 2 o más procesos. No existe una solución eficiente. El interbloqueo sucede cuando un proceso tiene un recurso y solicita el otro.

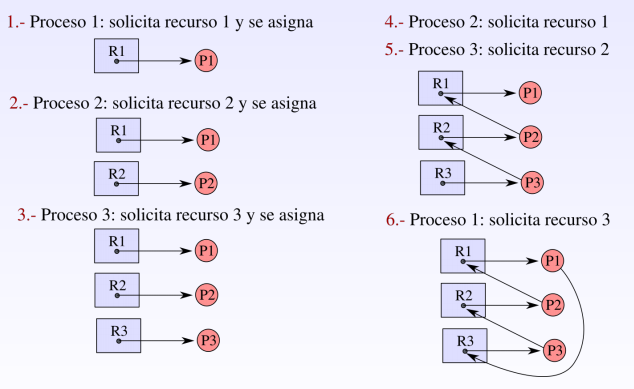
Existen **recursos reutilizables**, estos pueden ser usados por uno o más procesos y no se agotan con el uso. Los procesos obtienen unidades de recursos que liberan posteriormente para que otros procesos las reutilicen. Ejemplos de recursos reutilizables son procesadores, canales E/S, memoria principal y secundaria, archivos, bases de datos y semáforos.

Otro tipo de recurso son los **recursos consumibles**, estos pueden ser creados (producidos) y destruidos (consumidos) por un proceso. Ejemplos de recursos consumibles son interrupciones, señales, mensajes e información en buffers de E/S.

Grafos de asignación de recursos:



Ejemplo de grafo de asignación de recursos, con interbloqueo:



Las **condiciones** para que se produzca el interbloqueo son:

1. Exclusión mutua: Solo un proceso puede usar un recurso cada vez.
2. Retención y espera: Hay al menos un proceso que tiene asignado un recurso y se encuentra en espera de que otro proceso libere otro recurso.
3. No apropiación / Sin expropiación: No se puede forzar la expropiación de un recurso al proceso que lo tiene.
4. Espera circular: Si se dibuja el grafo, en este hay un ciclo.

Los interbloqueos se pueden **gestionar** de las siguientes formas:

1. No hacer nada (algoritmo del avestruz).
2. Uso de protocolos que aseguren que el sistema no se bloqueará:
    - **Prevención**: Garantizar que una (o más) de las condiciones necesarias para la formación de interbloqueos no se cumpla.
        - **Prevención de la exclusión mutua**: Hay recursos que necesariamente requieren la exclusión mutua, por lo tanto, prevenir la exclusión mutua es no es una solución.
        - **Prevención de la retención y espera**: Los procesos deben solicitar todos los recursos al comenzar su ejecución. Un proceso queda bloqueado hasta que se le conceden

simultáneamente todas sus solicitudes. El problema con esta solución es que se hace un uso ineficiente de los recursos y existe la posibilidad de que se produzca inanición.
    - **Prevención contra la condición de no apropiación / sin expropiación**: Si un proceso solicita un recurso no disponible, se interrumpe y además se le quitan todos los recursos que tenía asignados. El proceso se reinicia cuando se le pueden proporcionar todos los recursos que tenía más el que solicitó y no estaba disponible. Es práctico y realizable con recursos cuyo estado puede ser fácilmente almacenable/recuperable.
    - **Prevención de la espera circular**: Hacer que los recursos puedan utilizarse uno a uno. Numerar los recursos de modo que solo se puedan solicitar en un orden determinado.
- **Evasión (basada en predicción)**: Proporcionar al sistema información anticipada sobre las necesidades de recursos de los procesos, para que pueda predecir qué ocurrirá y encontrar secuencias de asignación de recursos que eviten los interbloqueos.

3. Permitir que el sistema se bloquee y proporcionar mecanismos de **detección periódica** y **recuperación** de interbloqueos. Se puede usar el **algoritmo del banquero** con el cual podemos saber si el estado actual es seguro.

- Vector **Recursos existentes** – *Exist* $E = \{E_1, E_2, E_3, ..., E_m\}$
- Vector **Recursos disponibles** – *Available* $A = \{A_1, A_2, A_3, ..., A_m\}$
- Matriz **Asignación actual** – *Current* $C = \begin{Bmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nm} \end{Bmatrix}$
- Matriz **Solicitudes** – *Required* $R = \begin{Bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ R_{21} & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nm} \end{Bmatrix}$

El sistema operativo debe encargarse de que los procesos se **recuperen** de los interbloqueos, tenemos varias estrategias:

1. Recuperación mediante apropiación: Se selecciona un proceso (o varios) y se les requisan los recursos para cederlos a otros procesos bloqueados.
2. Recuperación mediante Rollback: Se almacena periódicamente el estado del proceso y el estado de los recursos utilizados. Al producirse un bloqueo se detectan los recursos que son necesarios y un proceso que tenga alguno de esos recursos se interrumpe y se retrasa hasta el punto de verificación anterior a la solicitud del recurso.
3. Recuperación mediante eliminación de procesos: Hay 2 posibles actuaciones:
    a. Se abortan todos los procesos bloqueados.
    b. Se selecciona un proceso que tenga uno de los recursos necesarios para el bloqueo y que se puedan reiniciar. Normalmente existe un criterio para elegir que proceso abortar, menor prioridad, mayor tiempo restante…

Para poder **evadir** interbloqueos, antes de asignar un recurso hay que **predecir** qué ocurrirá si se asigna. Se simula que se concede el recurso y se aplica el algoritmo del banquero. Si se encuentra una secuencia de asignación de recursos factible que no conduzca al interbloqueo se concede el recurso.