Unidad 1:
Gestión de Procesos

Tema 1, Concurrencia:
Exclusión mutua y sincronización.

1.1 Problema de la sección crítica, alternativas al uso de semáforos:
- Regiones críticas, Monitores, Variables de condición y Paso de mensajes.

1.2 Problemas clásicos de sincronización:
- Productor consumidor usando monitores y paso de mensajes.
  Problema de los Lectores /escritores.

1.1 Problema de la sección crítica.

- El problema de la sección crítica (Conceptos generales):
  - **Concurrencia:** Existencia simultánea de varios procesos en ejecución.
  - **IMPORTANTE: EXISTENCIA SIMULTÁNEA NO IMPLICA EJECUCIÓN SIMULTÁNEA.**
  - Necesidad de sincronización y comunicación.
    - **Comunicación:** Necesidad de **transmisión de información** entre procesos concurrentes.
    - **Sincronización:** Necesidad de que las ejecuciones de los procesos concurrentes se produzcan según una **secuenciación temporal**, conocida y establecida entre los propios procesos.

1.1 Problema de la sección crítica.

- El problema de la sección crítica (Conceptos generales):
  - **Exclusión mutua:** Para que el **acceso a ciertos recursos sea exclusivo de un proceso cada vez**. A la parte del programa que los utiliza se le llama sección crítica.
  - **Sección crítica:**
    - Cada proceso tiene un segmento de código llamado sección crítica.
    - **No está permitido que varios procesos estén simultáneamente en su sección crítica.**
    - Un protocolo rige la forma de entrar y salir de la sección crítica.

1.1 Problema de la sección crítica.

- El problema de la sección crítica (Conceptos generales):
  - Cualquier solución al problema de la sección crítica debe satisfacer los tres requisitos:
    - **Exclusión Mutua:** Sólo un proceso ejecuta simultáneamente su sección crítica.
    - **Progreso:** Cuando ningún proceso ejecuta su sección crítica, algún proceso que lo solicite podrá entrar utilizando un protocolo, que impida la entrada simultánea de varios. La decisión de quién entra no se puede posponer indefinidamente.
    - **Espera limitada:** Ningún proceso debe esperar ilimitadamente la entrada en la sección crítica.

1.1 Problema de la sección crítica.

- Problemas al uso de semáforos:
  - **Semáforos:** mecanismo cómodo y efectivo para lograr la sincronización.
  - **Problemas:**
    - Pueden ocasionar **errores de temporización** difíciles de detectar, puesto que sólo se producen con ciertas secuencias de ejecución específicas.
    - El **uso desordenado de las primitivas** podría producir que **no se consiga la exclusión mutua**, ya que dos procesos podrían estar simultáneamente en sus secciones críticas.
    - La **construcción de grandes programas resulta muy árduo** porque las primitivas **wait()** y **signal()** se distribuyen por todo el código y no es fácil advertir su efecto global sobre el valor de los semáforos.
    - **Bloqueos mutuos**.

1.1 Problema de la sección crítica.

- Alternativas al uso de semáforos:
  - Región crítica.
  - Monitor y Variables condición.
  - Paso de mensajes.

1.1 Problema de la sección crítica.

- Región crítica (RC):
  - Mecanismo de **sincronización** de alto nivel.
  - Requiere declarar una **variable que puede ser compartida entre muchos procesos**, pero a la que sólo podrá acceder un único proceso, a través de un mecanismo booleano de control de acceso.
  - Si el número de procesos dentro de esa **RC es igual a 0, un proceso que lo desee puede entrar a dicha RC**.
  - Si el número de procesos dentro de una **RC es igual a 1** y N procesos quieren entrar, esos **N procesos deben esperar**.
  - Cuando un proceso sale de una RC se permite que entre uno de los procesos que esperan.

1.1 Problema de la sección crítica.

- Región crítica condicional (RCC):
  - Funcionamiento similar a la RC, sólo que además, para que un proceso ejecute su sección crítica **la condición de acceso debe de ser cierta.**
  - La evaluación de la condición de acceso se considera parte de la región crítica.
  - En caso de que el **resultado de la evaluación sea falso**, **abandona la RC** para permitir a otros procesos entrar en ella.
  - Un proceso que haya evaluado la condición a falso no vuelve a entrar a su RC hasta que otro proceso abandone ésta (**espera activa**):
    - Se vuelve a ejecutar cuando “posiblemente” alguien haya modificado dicha condición.

1.1 Problema de la sección crítica.

- Región crítica condicional (RCC):
  - Limitaciones de las RCC:
    - Aunque mejoran algunos aspectos negativos de los semáforos, tienen algunas limitaciones:
      - Pueden aparecer **a lo largo de todo el programa**.
      - No se garantiza la **integridad de las estructuras de datos** compartidas.
      - Realizar una **implementación eficiente** de las mismas es una **tarea difícil**.

1.1 Problema de la sección crítica.

- Monitor:
  - Mecanismo de **sincronización** de alto nivel.
  - Funcionalidad equivalente a la de los semáforos pero más fáciles de controlar.
  - Tipo abstracto de datos que conjuga:
    - **Estructuras de datos**.
    - Conjunto de **operaciones** asociadas a tales estructuras.

    +
    - **Exclusión mutua**
    - **Sincronización** (variables de condición).

1.1 Problema de la sección crítica.

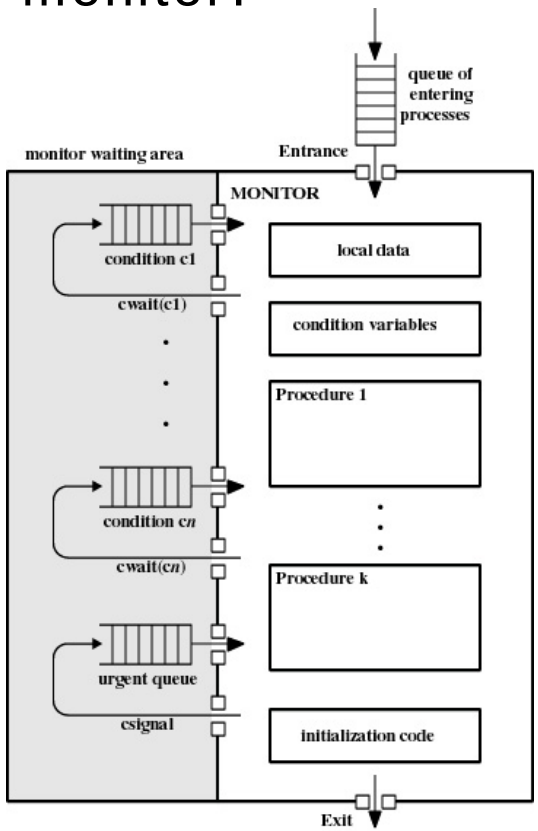
- Monitor:
  - Es un **módulo de software**:
    - Consta de **uno o varios procedimientos**.
    - **Secuencia de inicio.**
    - **Datos locales.**
  - Características básicas:
    - **Variables locales sólo accesibles para los procedimientos del monitor** y no para procedimientos externos.
    - Un **proceso entra** en el monitor **al invocar uno de sus procedimientos.**
    - **Sólo un proceso** se puede estar **ejecutando en el monitor** en un instante dado.

1.1 Problema de la sección crítica.

- Monitor:
  - Deben incluir **herramientas de sincronización**:
    - Un monitor proporciona sincronización por medio de las variables de condición que se incluyen dentro del monitor y que son accesibles sólo desde dentro.
  - Dos funciones operan con las variables de condición:
    - **cwait(c): suspende** (bloquea) la ejecución del proceso que llama a la condición “c”. El monitor estará disponible para ser usado por otro proceso.
    - **csignal(c): reanuda la ejecución de un proceso** que fue suspendido por un cwait(). Si hubiera varios elige uno de entre ellos y si no hay ninguno no hace nada.

1.1 Problema de la sección crítica.

- Estructura de un monitor:



1.1 Problema de la sección crítica.

- Monitor, Variables condición:
  - ¿Qué ocurre cuando un proceso P realiza una operación csignal() sobre una variable condición y existe un proceso suspendido Q asociado a dicha variable?
  - Si varios procesos están suspendidos por la concición c y algún proceso ejecuta csignal(c) ¿qué proceso se reanuda?.

1.1 Problema de la sección crítica.

- Soluciones al problema del productor/consumidor con buffer acotado:
  - Por medio de **Monitores con señales** (monitor de Hoare).
  - Por medio de **Monitores con notificación y difusión** (monitor de Lampson y Redell).

1.1 Problema de la sección crítica.

- Solución al P/C con monitores con señales:
  - El procedimiento **comprueba primero la condición "no_lleno", para saber si hay espacio libre en el buffer**. De no haberlo el proceso que está ejecutando el monitor se suspende ⇒ cualquier otro proceso (P o C) puede entrar ahora al monitor.
  - Cuando el **buffer ya no esté lleno el proceso suspendido puede ser retirado de la cola y reactivado** y el proceso podría reanudarse.
  - **Tras introducir un carácter en el buffer**, el proceso activa la condición **"no_vacío".**
  - **Conclusiones:** Como vemos **la propia estructura del monitor garantiza la EM**, no es posible que P y C accedan simultáneamente al buffer. Lo único que tiene que hacer el programador es situar correctamente las primitivas cwait() y csignal() en el monitor controlando que no se depositen elementos en un buffer lleno o no se extraigan de uno vacío.

1.1 Problema de la sección crítica.

- Inconvenientes de los monitores con señales:
  - Posible **error de sincronización de monitores**: Si olvidamos cualquiera de los csignal(), los procesos que entran en la cola de la condición se quedan colgados permanentemente.
  - Si un proceso ejecuta un csignal() y no ha terminado en el monitor, hacen falta **dos cambios de proceso adicionales** para continuar la ejecución del programa:
    - Uno para suspender el proceso.
    - Otro para reanudarlo cuando el monitor quede disponible.
  - **La planificación debe de ser muy fiable**. Cuando se ejecuta un csignal() se activará inmediatamente un proceso de la cola de la condición correspondiente y el planificador debe asegurarse de que ningún otro proceso entre al monitor antes de la activación.

1.1 Problema de la sección crítica.

- Soluciones al P/C con monitores de notificación y difusión:
  - **Notifican los procesos** en vez de reactivarlos a la fuerza.
  - Se puede añadir una **primitiva de difusión "cbroadcast()",** que provoca que todos los procesos que están esperando por una condición se sitúan en el estado de "listos". Esto es conveniente en situaciones donde un proceso no sabe cuantos procesos deben reactivarse.
  - **Ventajas sobre los monitores con señales**:
    - 1. Menor propensión a errores, cada procedimiento comprueba la variable del monitor después de ser despertado, por medio del while.
    - 2. Más modulable en cuanto a la construcción de programas.
    - 3. Hace cumplir la EM y concluye la operación de E/S antes de permitir cualquier otra operación sobre el buffer.
    - 4. Dispone de suficiente memoria para que este proceso pueda completar su solicitud de asignación.

1.1 Problema de la sección crítica.

- Paso de mensajes:
  - **Refuerzo de la Exclusión Mutua**.
  - **Intercambio** de información.
  - Se caracteriza por dos primitivas:
    - **send(destino, mensaje):** envía mensaje a proceso de destino.
    - **receive(origen, mensaje):** información indicando el proceso emisor y el mensaje.

1.1 Problema de la sección crítica.

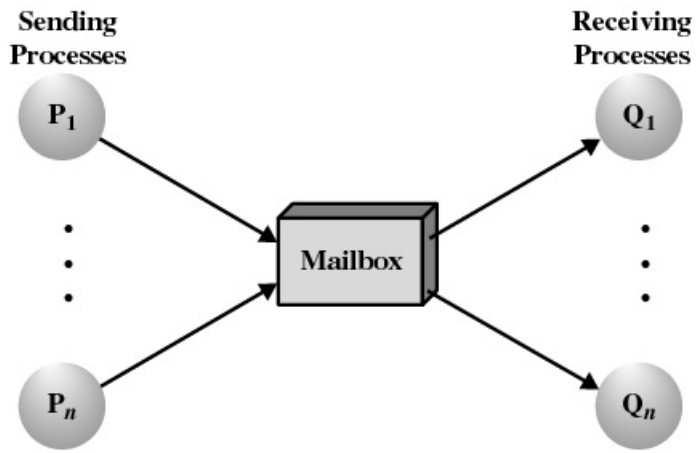
- Implementación de paso de mensajes:
  - **Sincronización:** Entre procesos, P1 no puede recibir un mensaje hasta que lo envíe P0.
    - Emisor y receptor pueden ser bloqueantes o no bloqueantes (esperando un mensaje). Posibles combinaciones:
      - **send() bloqueante y receive() bloqueante:**
        - Tanto el emisor como el receptor se bloquean hasta que se entrega el mensaje.
      - **send() no bloqueante y receive() bloqueante:**
        - Permite que un proceso envíe uno o más mensajes a varios destinos tan rápido como sea posible.
        - El receptor se bloquea hasta que llega el mensaje solicitado.
      - **send() no bloqueante y receive() no bloqueante:**
        - Ninguno debe esperar.

1.1 Problema de la sección crítica.

- Direccionamiento:
  - Debemos especificar en el send() que proceso va a recibir el mensaje, y también que el receive() conozca el origen del mensaje que va a recibir ⇒ dos esquemas:
    - **Direccionamiento directo:**
      - Send() incluye una identificación del proceso de destino.
      - Receive() puede conocer de antemano de qué proceso espera un mensaje.
      - Direccionamiento implícito: receive() puede utilizar el parámetro origen para devolver un valor cuando se haya realizado la operación de recepción.
    - **Direccionamiento indirecto:**
      - Los mensajes no van directamente del emisor al receptor.
      - Los mensajes se envían a una estructura de datos compartida formada por colas (**buzones o mailboxes**).
      - P0 envía mensajes al buzón apropiado y P1 los recoge del buzón.

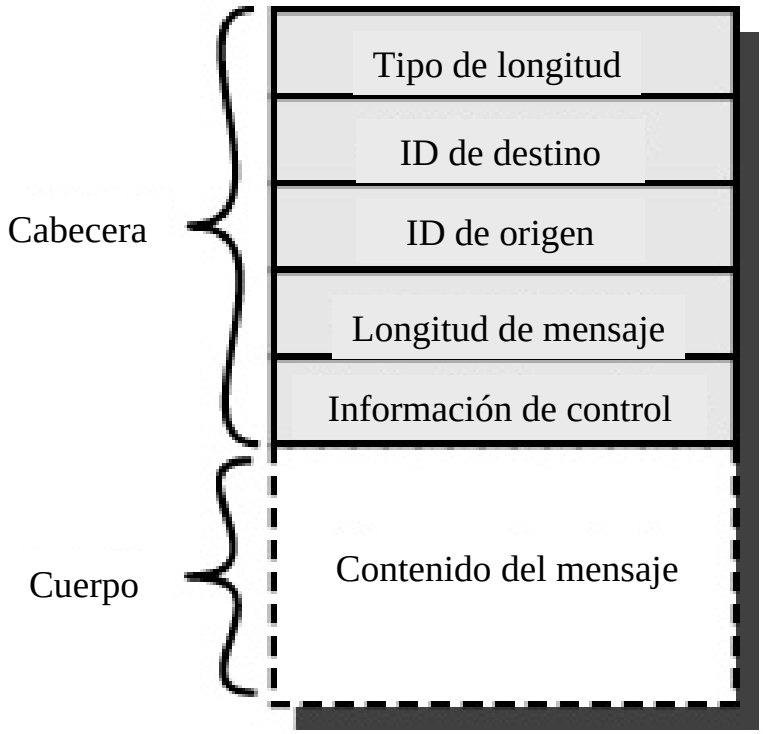
1.1 Problema de la sección crítica.

- Relaciones de direccionamiento:
  - **Uno a uno:** enlace privado de comunicación entre P0 y P1.
  - **Uno a varios:** aplicaciones en los que el mensaje se difunde a un conjunto de procesos.
  - **Varios a uno:** interacciones cliente/ servidor, un proceso ofrece un servicio a un conjunto de procesos.
  - **Varios a varios:**



1.1 Problema de la sección crítica.

- Formato de mensajes:



1.1 Problema de la sección crítica.

- Implementación de la EM mediante paso de mensajes:

```
/* programa exclusion mutua */
const int n= /*numero de procesos*/;
void P(int i){
    mensaje msj;
    while (cierto){
        receive (exmut, msj);
        /* sección crítica */
        send (exmut, msj);
        /*resto*/
    }
}
```

```
void main(){
    crear_buzon(excmut);
    send(excmut, null);
    parbegin(P(1),P(2), ..., P(n));
}
```

1.1 Problema de la sección crítica.

- Implementación de la EM mediante paso de mensajes:
  - Un proceso (P) que desea entrar en su sección crítica **intenta primero el receive(), si el buzón está vacío P se bloquea**. Si el proceso consigue el mensaje, ejecuta su sección crítica y después devuelve el mensaje al buzón ⇒ el mensaje funciona como un testigo que se pasa de un proceso a otro.
  - Esta técnica implica que si hay **varios procesos ejecutando simultáneamente el receive():**
    - Si hay un mensaje, se entrega sólo a uno de los procesos y los otros se bloquean.
    - Si el buzón está vacío, todos los procesos se bloquean. Cuando haya un mensaje disponible sólo se activará y tomará el mensaje uno de los procesos bloqueados.

1.2 Problema de los lectores/escritores.

- Enunciado:
  - Tenemos un **área de datos** (que puede ser un archivo, un bloque de memoria principal o un banco de registros del procesador) **compartida entre varios procesos**. Algunos procesos sólo leen los datos (lectores) y otros sólo escriben (escritores). Se puede satisfacer que:
    - 1. Cualquier número de lectores puede leer el archivo simultáneamente.
    - 2. Sólo puede escribir en el archivo un único escritor en cada instante.
    - 3. Si un escritor está accediendo al archivo, ningún lector puede leerlo.