Sistemas Operativos

UNIDAD 3

EXCLUSIÓN MUTUA Y SINCRONIZACIÓN MEDIANTE COLAS DE MENSAJES

Exclusión Mutua

SOLUCIONES SOPORTE SO

PASO DE MENSAJES – COLAS DE MENSAJES

Exclusión Mutua: MENSAJES

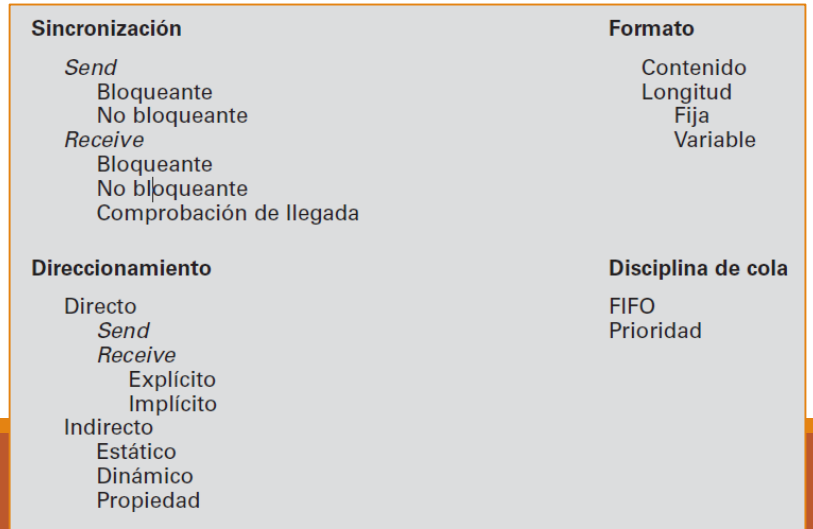
Los semáforos y monitores:
- Un único computador (mono o multiprocesador), compartiendo memoria

El paso de mensajes, además:
- Sistema distribuido (conectados en red).

Intercambio de información. Primitivas:
- **send (destino, mensaje)**
- **receive (origen, mensaje)**

Aspectos de diseño
- Acuse de recibo. Perdida del mensaje. Perdida del acuse. Secuenciación de mensajes.
- Verificación de autenticidad.
- Rendimiento. En la misma máquina puede ser lento el mecanismo de mensajes frente a monitores o semáforos.

Análisis desde las siguientes perspectivas
- Sincronización
- Direccionamiento
- Formato
- Disciplina de cola



Exclusión Mutua: MENSAJES

Sincronización

- Primitivas bloqueantes o no bloqueantes
  - Envío **bloqueante**, recepción **bloqueante** (rendez-vouz)

    Permite sincronización estricta entre varios procesos
  - ➢ Envío **no bloqueante**, recepción **bloqueante**

    Combinación más común

    Requiere de mensajes de respuesta (ACK) si se quiere verificar recepción
  - Envío **no bloqueante**, recepción **no bloqueante**

Posibles problemas:

- Un envío no bloqueante tiene el peligro de reenvíos continuos sin bloqueo del emisor.
- Una recepción bloqueante puede ser indefinida si el mensaje del emisor se pierde.

Posibles soluciones:

- Se puede permitir que un proceso compruebe la existencia del mensaje por el que espera antes de solicitarlo.
- Esto es útil si puede haber varios emisores.

Exclusión Mutua: MENSAJES

Direccionamiento

- Direccionamiento Directo.
  - Las primitivas incluyen identificador del proceso con el que comunicar
  - Requiere el conocimiento de los procesos origen y destino
- Direccionamiento Indirecto.
  - Se usan Buzones para intercambiar los mensajes
    - Este esquema desacopla emisor y receptor
    - Los buzones pueden ser
      - Estáticos: El buzón se crea y se asigna a un proceso. Al morir el proceso de destruye el buzón.
      - Dinámicos: Asociación del proceso a un buzón de manera dinámica con primitivas *conectar/desconectar*
    - El propietario del buzón puede ser:
      - El emisor.
      - El receptor.

      } Se crea el buzón y se asigna al emisor/receptor.
      Se destruye cuando finaliza el emisor o mediante primitivas.
      - El Sistema operativo. En este caso se necesitan primitivas de creación y destrucción del buzón.

Exclusión Mutua: MENSAJES

- Direccionamiento Indirecto

  Relación entre emisores/receptores

  - Uno a Uno

    Permite un enlace privado de comunicaciones entre dos procesos

  - Uno a Muchos

    Útil en aplicaciones donde es necesario difundir un mensaje a N procesos (broadcast)

  - Muchos a Uno

    El buzón se denomina puerto.

    El receptor (servidor) ofrece un servicio a varios procesos.

    El puerto está disposición de los emisores (clientes) para recibir peticiones de servicio.

  - Muchos a Muchos

    Permite a múltiples procesos servidores proporcionar un servicio concurrente a múltiples clientes.

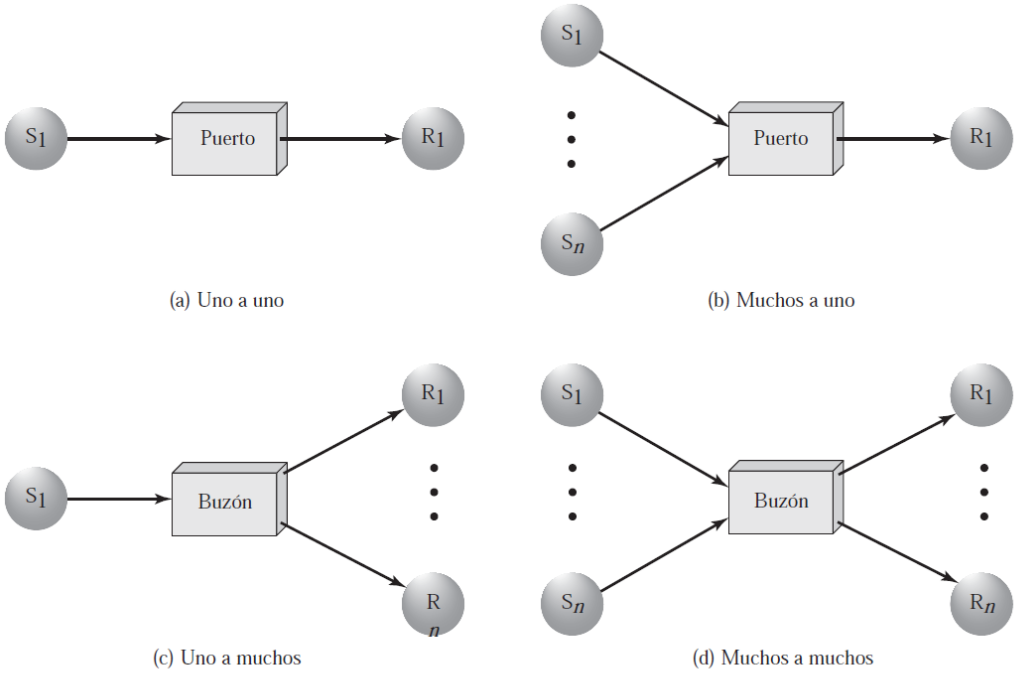
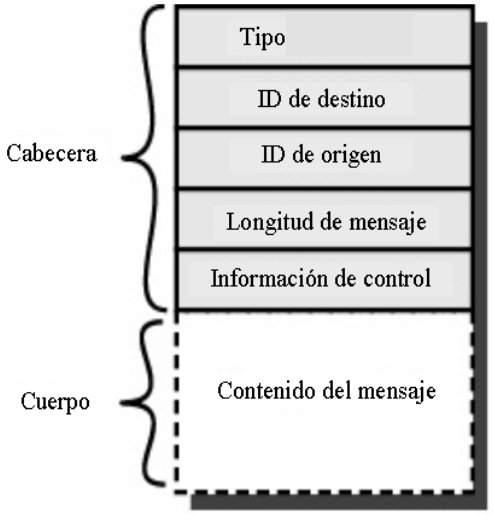


Figura 5.18. Comunicación indirecta de procesos.

Exclusión Mutua: MENSAJES

Formato.

- Depende del servicio de mensajería si es sobre un sistema independiente o distribuido
- Mensajes cortos de tamaño fijo:
  - Minimizan el almacenamiento y se pude transmitir el mensaje.
- Mensajes largos
  - Los datos se ubican en un fichero y se transmite su referencia.
- Mensajes de longitud variable. Se divide en dos partes
  - Cabecera:
    - Tipo de mensaje
    - Proceso destino (buzón, puerto)
    - Proceso origen (cliente)
    - Longitud del mensaje
    - Información de control (prioridad, siguiente mensaje, contador, etc…)
  - Cuerpo: Contenido del mensaje



Disciplina de la cola

- FIFO: Puede no ser suficiente si hay mensajes mas urgentes que otros.
- PRIORIDADES: En función del tipo de mensaje, del destinatario o emisor.
- SELECCIÓN: Permitir que el receptor examine la cola en busca del mensaje que quire recibir a continuación.

Exclusión Mutua: MENSAJES

Exclusión mutua con Mensajes

- send **no bloqueante** y receive **bloqueante**.

```
/*program exclusión mútua*/
const int n = /*Número de procesos*/

void P(int i){
        mensaje msj;
        while true {
                receive(mutex, msj);
                /*sección crítica*/
                send(mutex, msj);
                /*resto*/
                }
}

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

Exclusión Mutua: MENSAJES

Ejercicio:

- Resolver el problema del productor/consumidor con buffer **ilimitado** usando mensajes

Solución: *send no bloqueante, receive bloqueante*

```
/*program Productor Consumidor Buffer ilimitado*/
msg_t /*tipo abstracto que define el item, es el mensaje a pasar */

void Productor(){
     msg_t item;
     while true {
          item = producir();
          send(buffer,item);
          }
}

void Consumidor(){
     msg_t item;
     while true {
          receive(buffer,item);
          consumir(item)
          }
}

void main(){
     crear_buzon(buffer);
     parbegin(Productor(), Consumidor());
}
```

Exclusión Mutua: MENSAJES

Ejercicio:

Resolver el problema del productor/consumidor con buffer **limitado** por medio de mensajes.

Solución: *send no bloqueante, receive bloqueante*

```
/*program Productor Consumidor Buffer ilimitado*/
const int capacidad = N;
msg_t null;  /*mensaje vacío*/
int i;

void Productor(){
     msg_t msjp;
     while true {
          receive(buffer_producir,msjp);
          msjp=producir();
          send(buffer_consumir,item);
          }
}

void Consumidor(){
     msg_t msjc;
     while true {
          receive(buffer_consumir,msjc);
          consumir(msjc);
          send(buffer_producir,null);
          }
}

void main(){
     crear_buzon(buffer_producir);
     crear_buzon(buffer_consumir);
     for (int i=1; i<=capacidad; i++)
          send(buffer_producir,null);
     parbegin(Productor(), Consumidor());
}
```

Exclusión Mutua: MENSAJES

Ejercicio:

Implementar una barrera para un numero arbitrario (n) de procesos utilizando primitivas genéricas de paso de mensajes *send* y *receive*

Explicar detalladamente la solución alcanzada.

Paso de mensajes en Unix

Unix System V Release4 (SVR4) proporciona tres herramientas para el paso de mensajes

- Tuberías
- Colas de Mensajes

} En la misma máquina

- Sockets → En distinta máquina

Tuberías

- Tuberías sin nombre: Para comunicar procesos con relación de parentesco
- Tuberías con nombre: Para comunicar procesos sin relación de parentesco

Colas de Mensajes

- Buffer FIFO mantenidos por el S.O. donde puede haber varios procesos escribiendo y varios leyendo.
- La exclusión mutua al buffer está garantizada por el S.O.
- Se pueden escribir o leer estructuras con formato de mensaje tratados (leidos/escritos) como items indivisibles
- Se pueen tipar los mensajes y dotar de primitivas de selección de mensaje por tipo (esto rompe FIFO, aunque FIFO para mismo tipo)
- Primitivas principales:
  - **msgget** : Para crear una cola o habilitar acceso a una cola ya creada
  - **msgctl** : Control de colas de mensajes
  - **msgsnd**: Para escribir (send) mensaje en la cola
  - **msgrcv**: Para leer (receive) mensaje de la cola

Fin

UNIDAD 3

EXCLUSIÓN MUTUA Y SINCRONIZACIÓN CON MONITORES