

```

begin
  parbegin
    producer;
    consumer;
  parend
end.

```

Debido a las limitaciones de espacio que afectan a la elaboración de un temario de este tipo remitimos al lector que desee consultar un análisis más detenido de los monitores a la bibliografía señalada y, en particular, a [Deitel93] y [Tanenb93].

2.3. COMUNICACIÓN ENTRE PROCESOS EN SISTEMAS DISTRIBUIDOS

En la década de los noventa se han hecho comunes los **sistemas distribuidos** con varias CPU, cada una con su propia memoria y otros recursos, unidas por un sistema de red. En estos sistemas ya no son aplicables los semáforos, los contadores de eventos o los monitores para permitir que procesos que se ejecutan en computadoras diferentes (y que por tanto no comparten memoria) se comuniquen entre sí. Por ello ha sido necesario acudir a otros esquemas de comunicación entre procesos más elaborados como los que presentamos en los siguientes apartados.

2.3.1. INTERCAMBIO DE MENSAJES

2.3.1.1. Concepto

La forma general de enviar y recibir mensajes es usando las siguientes primitivas del sistema operativo:

```

enviar (destinatario, mensaje);
recibir (origen, mensaje);

```

Debe quedar claro que se trata de llamadas al sistema y no comandos del lenguaje por lo que son accesibles desde muchos entornos de lenguajes de programación.

Un **envío con bloqueo** debe esperar hasta que el receptor reciba el mensaje constituyendo un ejemplo de comunicación síncrona. Un **envío sin bloqueo** permite al transmisor continuar con otras tareas aunque el receptor no haya recibido aún el mensaje permitiendo pues que ambos interlocutores se comuniquen asincrónicamente; los envíos sin bloqueos requieren de la utilización de buffers para almacenar los mensajes hasta que los reciba el receptor. El destinatario puede ser un proceso concreto, todos los procesos o bien algún grupo concreto de procesos.

La comunicación asíncrona aumenta las posibilidades de paralelismo. Por ejemplo, un procesador de textos puede enviar un documento para imprimir a un gestor de impresora muy ocupado; el sistema almacena automáticamente la información hasta que el gestor de impresora esté listo para recibirla y el transmisor continuará su ejecución sin tener que esperar ser atendido por aquel.

Mientras no se reciba un mensaje una llamada a recibir con bloqueo obligará al receptor a esperar; una llamada a recibir sin bloqueo permite que el receptor continúe realizando tareas antes de volver a intentar la recepción. Al igual que para la emisión, el origen de un mensaje puede ser un proceso determinado, cualquier proceso o uno que pertenezca a un grupo concreto de procesos.

En los sistemas distribuidos la transmisión puede tener errores e incluso perderse. Por ello, los transmisores y receptores han de usar un protocolo de reconocimiento para confirmar la recepción correcta de cada transmisión. No obstante, el estudio de este aspecto de la comunicación entre procesos cae ya dentro de los temas dedicados a las comunicaciones.

Los sistemas con mensajes deben también enfrentarse a la necesidad de dar nombres en forma no ambigua a los procesos origen o destino de un mensaje. Un método muy usado es asegurar que cada computadora asigne nombres únicos a sus propios procesos utilizando para ello el propio nombre del ordenador en un esquema de nombres del tipo proceso@máquina. Ahora bien, si el número de máquinas es grande, podría darse el caso de que dos máquinas se asignen el mismo nombre. Por ello, se requiere un control centralizado para determinar un nombre

único para cada computadora de un sistema distribuido, lo cual no supone una sobrecarga significativa, pues los ordenadores se añaden o eliminan de las redes con relativa baja frecuencia. Con el fin de organizar de algún modo los grupos de máquinas afines y con ello reducir la posibilidad de conflictos se usan los **dominios** o agrupaciones de máquinas. De este modo, es posible dirigirse a los procesos como proceso@máquina.dominio. En este esquema tenemos ya tres niveles:

- El nombre de los procesos ha de ser diferente dentro de una máquina.
- El nombre de las máquinas ha de ser diferente dentro de un dominio.
- El nombre de los dominios ha de ser diferente.

Ahora bien, con este esquema distintas máquinas pueden tener procesos homónimos y distintos dominios pueden incluir máquinas con el mismo nombre.

Tal y como se estudia en otros temas del programa, las comunicaciones a base de mensajes en los sistemas distribuidos presentan serios problemas de seguridad. Uno de ellos es el problema la **autenticación** ¿Cómo saben los transmisores y los receptores que se están comunicando con interlocutores válidos y no con impostores que pretenden *hackear* su sistema?

2.3.1.2. Buzones y puertos

Las comunicaciones a base de mensajes se pueden establecer directamente entre un proceso transmisor y un proceso receptor los cuales deberán tener asignadas direcciones únicas. Un método alternativo consiste en utilizar una nueva estructura de datos denominada **buzón** consistente en un buffer en el que se almacenarán los mensajes enviados antes de ser extraídos por el receptor. En este caso caso, el transmisor y el receptor especifican el buzón, en lugar del proceso, con el que se van a comunicar.

Así pues, un buzón actúa como una cola de mensajes que puede ser utilizada por múltiples transmisores y receptores. En un sistema distribuido, los receptores de un mismo mensaje pueden estar situados en muchas computadoras diferentes. Por ello, en la mayoría de los casos, si queremos usar un envío único, se requieren dos transmisiones para los mensajes: una del transmisor a los buzones y otra de éstos a los respectivos receptores. Para agilizar el proceso se puede dar a un receptor un **puerto**; éste se define como un buzón utilizado por múltiples transmisores y un único receptor. En los sistemas cliente/servidor, cada servidor tiene un puerto asignado por el cual recibe los mensajes de multitud de clientes.

2.3.1.3. Conductos (pipes)

UNIX introdujo la noción de conductos (pipes) como un esquema para manejar comunicaciones entre procesos. Un conducto es en esencia un buzón que permite extraer un número especificado de bytes a la vez. El conducto no mantiene las fronteras entre mensajes. Si los procesos convienen en leer y escribir mensajes con un tamaño determinado, o terminar cada mensaje con un señalador especial (por ejemplo retorno de carro o nulo), la utilización de los tubos como vehículos de los mensajes deja de tener problemas.

Los conductos se estudian en más detalle en el estudio de la gestión de procesos en UNIX.

2.3.2. LLAMADAS A PROCEDIMIENTOS REMOTOS (RPC)

La noción de llamadas a procedimientos remotos (remote procedure calls, RPC) fue introducida por Birrel y Nelson en 1984 con el objeto de ofrecer un mecanismo estructurado de alto nivel para realizar la comunicación entre procesos en sistemas distribuidos. Con una llamada a un procedimiento remoto, un proceso de un sistema A llama a un procedimiento de un proceso de otro sistema B. El proceso de A se bloquea esperando el retorno desde el procedimiento llamado en el sistema remoto y después continúa su ejecución desde el punto que sigue inmediatamente a la llamada.

El procedimiento llamado y el que llama residen en procesos distintos, con espacios de direcciones distintos, por lo cual no existe la noción de variables globales compartidas, como en los procedimientos normales dentro de un solo proceso. Por tanto, las RPC transfieren la información estrictamente a través de los parámetros de la llamada.

Las llamadas a procedimientos remotos presentan una serie de problemas. En teoría, desde el punto de vista del usuario, una RPC debería ser indistinguible de otra a un procedimiento local. No obstante, y como ejemplo de las