Tema 4. Comunicación entre Procesos

Jorge García Duque
http://www.det.uvigo.es/~jgd
Depto. Enxeñería Telemática
Universidad de Vigo

Introducción

- Dos mecanismos básicos de comunicación entre procesos:
  - Memoria Compartida.
    - Necesitan un mecanismo de sincronización externo.
    - La responsabilidad de la comunicación recae en los procesos (el sistema operativo sólo proporciona llamadas para manipular dicha memoria compartida).
  - Paso de Mensajes.
    - La responsabilidad de la comunicación y sincronización recae en el Sistema Operativo que proporciona un enlace lógico entre procesos.
    - Los procesos sólo tienen que invocar correctamente a dos llamadas básicas: *send* y *receive* (bloqueantes o no).

Paso de Mensajes (I). Tipos de Comunicación

- *Comunicación Directa*: se nombran explícitamente los procesos.
  - *Direccionamiento simétrico*: se nombran el destino y el origen:
    - *P: send(Q,msg);*
    - *Q: receive(P,&msg);*
  - *Direccionamiento asimétrico*: se nombra sólo el destino:
    - *P: send(Q,msg);*
    - *Q: receive(&id_proc,&msg);*
- *Comunicación Indirecta:* se utilizan *buzones.*
    - *send(id_bz,msg);*
    - *receive(id_bz,&msg);*

Paso de Mensajes (II). Comunicación Directa

- Características de la *Comunicación Directa:*
  - El enlace lógico es implícito entre cada par de procesos.
  - Un enlace está asociado sólo a dos procesos.
  - Entre cada par de procesos existe un único enlace.
  - El enlace es bidireccional.
  - No compilación independiente.

Paso de Mensajes (III). Comunicación Indirecta

- Características de la *Comunicación Indirecta:*
  - El enlace lógico existe sólo si se comparte un buzón.
  - Un enlace puede asociarse a más de dos procesos.
  - Dos procesos pueden compartir varios enlaces.
  - El enlace puede ser unidireccional o bidireccional.
  - Compilación independiente.

¿Qué sucede si hay varios procesos esperando a recibir un mensaje del mismo buzón?.

Paso de Mensajes (IV). Capacidad del Enlace

- Capacidad del Enlace:
  - Capacidad 0: los procesos deben estar sincronizados.
  - Capacidad N: la sincronización sólo es necesaria si el enlace está lleno.
  - Capacidad ilimitada: el emisor nunca tienen que esperar.

¿Puede el emisor estar seguro de que su mensaje se ha entregado?

Llamada a Procedimiento Remoto (RPC) I

- Mecanismo de comunicación mediante el cual un proceso puede invocar un procedimiento que se ejecuta en un proceso remoto (en otro ordenador).
- Funcionamiento transparente como si fuera una llamada a un procedimiento local.
- Proceso cliente:
  - Invoca el procedimiento como uno local.
  - Un *stub* empaqueta los parámetros y los serializa conforme a un formato común para ser transmitidos.
  - Un módulo de comunicaciones se encarga del envío (petición identificando el procedimiento y parámetros) y espera los resultados.
  - El *stub* desempaqueta los resultados recibidos y los retorna.
- Proceso servidor:
  - Registra los procedimientos que pueden ser invocados remotamente.
  - El módulo de comunicaciones se encarga de recibir las peticiones y enviar los resultados.
  - El *stub* desempaqueta los parámetros, invoca al procedimiento y empaqueta los resultados.

Llamada a Procedimiento Remoto (RPC) II

- Problemas a tener en cuenta (respecto a una llamada a procedimiento local (LPC)):
  - Errores en la red o en el servidor: deben notificarse al cliente.
  - Semántica de ejecución:
    - En LPC: Semántica *exactamente-una-vez.*
    - En RPC:
      - Semántica *tal-vez.*
      - Semántica *al-menos-una-vez.*
      - Semántica *como-máximo-una-vez.*
  - En la ejecución del procedimiento no se tiene acceso al espacio de direccionamiento del cliente:
    - No se pueden utilizar variables globales.
    - No se pueden pasar punteros como parámetros.
  - Sobrecarga de procesamiento:
    - Empaquetado/desempaquetado de parámetros y resultados.
  - Sobrecarga de transferencia por la red.

Llamada a Procedimiento Remoto (RPC) III

- Ejemplos de entornos RPC:
  - *Sun-RPC (ONC-RPC: Open Network Computing-RPC)*: el más extendido en entornos Linux.
  - *DCE/RPC (Distributed Computing Environment RPC)*: RPC definido por la *Open Software Foundation*, similar al anterior.
  - *Java-RMI (Remote Method Invocation)*: añadido por Sun en 1995.
  - *CORBA (Common Object Requesting Broker Architecture)*: arquitectura para el intercambio de objetos en entornos distribuidos.
  - *SOAP (Simple Object Access Protocol)*: evolución de *XML-RPC,* utilizado en servicios web.
  - *DCOM (Distributed Component Object Model)*: desarrollado por Microsoft, basado en elementos *DCE/RPC.*
  - *.NET Remoting*: Infraestructura de invocacióon remota en entornos .NET