Tema 7: Paso de Mensajes

Elvira Albert

Departamento de Sistemas Informáticos y Computación

Universidad Complutense de Madrid

elvira@sip.ucm.es

Madrid, Mayo, 2021

Introducción

Paradigmas:

- *Procesadores con memoria compartida*: Las instrucciones de sincronización vistas hasta ahora se basan en L/E en variables compartidas.
- *Arquitecturas con memoria distribuidas*: los procesadores tienen su propia memoria y se comunican utilizando una red de interconexión intercambiando mensajes

Programación para arquitecturas con memoria distribuidas:

- Si utilizamos read & write sobre canal de comunicación necesitaríamos algo similar a sincronización con espera activa
- *Primitivas de paso de mensajes*
  - operaciones de red que incluyen sincronización
  - similares a los semáforos con datos
  - canal $\equiv$ abstracción de la red de comunicación

Sólo para arquitecturas distribuidas?

- Los procesos se pueden distribuir entre procesadores en arquitectura de memoria distribuida
- Los canales se pueden implementar utilizando memoria compartida para arquitecturas multi-procesador con memoria compartida

Conceptos básicos

- Los canales son los únicos recursos compartidos
- La capacidad de los canales es ilimitada
- Las variables son locales (sincronización no es necesaria)
- Los procesos deben comunicarse para interactuar

Mecanismos para programación distribuida

- Dos formas de nombrar y utilizar los caminos:
  - flujo de comunicación 1-dirección
  - flujo de comunicación 2-direcciones
- Dos formas de sincronizar la comunicación:
  - asíncrona (no bloqueante)
  - síncrona (bloqueante)
- Las combinaciones dan lugar a esto cuatro paradigmas:
  - paso de mensajes asíncronos (1-dirección)
  - paso de mensajes síncronos (1-dirección)
  - RPC: llamadas a procedimiento remoto (2-direcciones)
  - rendezvous (2-direcciones)
- Los 4 mecanismos son equivalentes (traducción de uno a otro es posible)

Contenidos Tema 7

7.1 Paso de mensajes asíncrono

Notación

- Declaración de un canal

  `chan ch(type1 id1,...,typen idn)`

  array de canales: `chan result[n](int i)`

- Envío no bloqueante

  `send ch(expr1,...,exprm)`

  evalúa las expresiones, crea un mensaje con los valores y lo coloca al final de la cola

- Recepción bloqueante

  `receive ch(var1,...,varm)`

  se extrae el mensaje en la cabeza de la cola y se asignan los valores a las variables

7.1 Paso de mensajes asíncrono

Comprobación vacío:

- Comprobación vacío: `empty(ch)`
- puede permitir que el proceso haga otra cosa si la cola está vacía
- puede devolver true y cuando empiece la ejecución de otro código llegar mensaje
- puede devolver false y en el momento de leerlo que esté vacío (si hay varios procesos leyendo)

Asumimos:

- el acceso al canal es atómico
- la entrega de mensajes es fiable y sin errores
- los mensajes se entregan en el orden en el que se añaden a la cola
- los canales son globales a todos los procesos