Clase 6- pasaje de mensajes asincrónicos

jueves, 19 de septiembre de 2024 13:17

Un programa distribuido es un programa concurrente que se comunica por mensajes. No tenemos problemas de variables pq no existen variables compartidas.

Solo va a haber procesos. Todas las variables pertenecen a procesos.

Procesos comparten canales físicos o lógicos que son la única cosa que comparten los procesos. La comunicación necesitan participación de ambs proceos. Uno dice yo lo mando a x y el otro dice yo recibo de y.
Es más difícil hacer que corra pero una vez que lo hizo no tengo problemas.

- Mailbox, input port, link.
- Uni o bidireccionales.
- Sincrónicos o asincrónicos. Sincrónico:: me demoro hasta que se reciba el mensaje. Asincronico se usa una especie de buffer. El que manda el mensaje lo deja ahí y punto

En pasaje asincrónico se puede simular sincronismo.

Pasajes de mensajes son unidireccionales, los otros dos bidireccionales
- Pasaje de Mensajes Asincrónicos (PMA)
- Pasaje de Mensajes Sincrónico (PMS)
- **Llamado a Procedimientos Remotos (RPC) --> solo en la teoria**
- Rendezvous

La sincronización de la comunicación interproceso depende del patrón de interacción:
- Productores y consumidores (Filtros o pipes)
- Clientes y servidores-- necesita comunicación bidireccional
- Pares que interactúan-- muchos procesos copia del mismo y cada tanto interactuan

Pasaje de mensajes asincrónicos

Tipo mailbox-- cualquiera deja y cualquiera recibe
Declaración de canales de forma explicita

Cada canal es una cola de mensajes mandados y todavia no recibidos
- chan entrada (char);
- chan acceso_disco (INT cilindro, INT bloque, INT cant, CHAR* buffer);
- chan resultado[n] (INT);

Operación Send un proceso agrega un mensaje al final de la cola ("ilimitada") de un canal ejecutando un send, que no bloquea al emisor:
send ch(expr1, ... , exprn);
Operación Receive un proceso recibe un mensaje desde un canal con receive, que demora ("bloquea") al receptor hasta que en el canal haya al menos un mensaje; luego toma el primero y lo almacena en variables locales:

receive ch(var1, ... , varn);
El tipo de comunicación no es bloqueante, se puede bloquear.
Se queda dormido hasta que llega un mensaje, no consume proce mientras

Cada dato del canal es para una operación. Si 2 quieren mandar/recibir por un canal se hace de a uno por vez operaciones atómicas.
Respeta orden fifo(cola)

empty(ch) determina si la cola de un canal está vacía. Útil cuando el proceso puede hacer trabajo productivo mientras espera un mensaje, pero debe usarse con cuidado.
O podría ser false, y no haber más mensajes cuando sigue ejecutando (si no en el único en recibir por ese canal).
La evaluación de empty podría ser true, y sin embargo existir un mensaje al momento de que el proceso reanuda la ejecución.

Cuando yo hago receive me quedo tieso hasta que recibo, con empty capa veo que está opcuapdo y voy a hacer otra cosa

Empty si tengo solo un proceso que hace receive sobre ese canal está bien, pero hay problema si el canal tiene más de un receptor pq capaz yo pregunto ahora y entre que yo pregunté e hice el receive alguien más tomó el dato entonces me quedo demorado

Mailbox: cualquier proceso manda y recibe
Input port: un solo receptor y muchos emisores
Link: único emisor, único receptor

Productores y consumidores filtro

Proceso recibe mensajes de canales de entrada y manda a canaes de salida habiendo aplicado algo. Se va pasando por los procesos hasta que se termina de filtrar.

Clientes y Servidores. Monitores Activos

Monitor -> manejador de recurso. Encapsula variables permanentes que registran el estado, y provee un conjunto de procedures. Los simulamos, usando procesos servidores y PM, como procesos activos en lugar de como conjuntos pasivos de procedures.

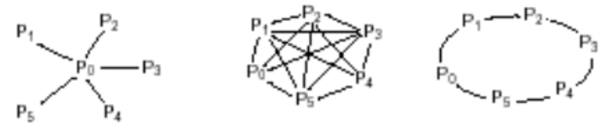
Servidor(monitor) deberlía estar en un core separado. maneja pedidos de clientes.

Cliente manda mensaje a un canal de requerimientos general, recibe en canal propio

Zzzzzzzzzzzz ver que onda todo lo del medio

Peers interactuantes, cambio de valores

Cada proces tiene un dato V y tods los procesos tiene que conocer el menor y el mayor



(a) Centralized solution (b) Symmetric solution (c) Ring solution