Tema 5: Monitores

Elvira Albert

Departamento de Sistemas Informáticos y Computación

Universidad Complutense de Madrid

elvira@sip.ucm.es

Madrid, Abril, 2021

Introducción

Semáforos: mecanismo de bajo nivel

- propenso a errores
  - olvidar P o V accidentalmente, demasiados P o V
  - utilizar semáforo equivocado, no proteger una sección crítica
- globales a todos los procesos
- se implementa **exclusión mutua** y **sincronización condicional** de la misma forma
  - dificulta la comprensión

Monitores: mecanismo de abstracción de datos de alto nivel

- encapsula la representación de un objeto abstracto
- proporciona operaciones para manipular los datos
  - acceso a las variables llamando a las operaciones del monitor
  - **exclusión mutua** aseguran que los procedimientos de un mismo monitor se ejecutan en exclusión mutua
  - **sincronización condicional** a través de variables de condición

Contenidos Tema 5

5.1 Sintaxis y semántica

5.2 Técnicas de sincronización

- bounded buffers
- readers and writers
- shortest-job-next
- temporizador de intervalos
- sleeping barber

5.1 Sintaxis y semántica

Monitor $\equiv$ clase que agrupa la representación e implementación de un recurso compartido:

- declaración de variables que representan estado del recurso
- procedimientos que implementan las operaciones

Sintaxis:

```
monitor mname{
  declaración variables;
  inicialización;
  procedimientos;
}
```

Propiedades:

1. sólo los nombres de procedimientos son visibles;
   `call mname.opname(args)`
2. los procedimientos del monitor no pueden acceder a variables externas al monitor;
3. las variables se inicializan en la creación del monitor

Invariante del monitor:

- predicado que deben satisfacer los estados de las variables del monitor cuando ningún proceso está accediendo a ellas
- el código de inicialización establece el invariante
- todos los procesos lo preservan

Exclusión mutua implícita:

- los procedimientos del monitor por definición ejecutan en exclusión mutua
- no hay interferencia, no son necesarios protocolos de entrada y salida explícitos a las secciones críticas

Variables de condición:

- retrasar a un proceso hasta que se cumpla condición booleana
- **cond cv**: el valor de **cv** es la cola de procesos atrasados
- **empty(cv)**: pregunta el estado de la cola
- **wait(cv)**: el proceso se pone en la cola y puede entrar otro en el monitor (deja el mutex)
- **signal(cv)**: despierta al proceso en la cabeza de la cola **cv**, si la cola está vacía no tiene efecto
- **wait/notify**: la implementación de las colas suele ser FIFO, los procesos se retrasan en el orden en que llaman a wait y se despiertan en el orden en el que fueron retrasados

Disciplinas de señalización: proceso ejecuta signal(cv) $\rightarrow$ despierta proceso wait(cv) $\rightarrow$ quién ejecuta?

- **SC** (signal and continue): el señalizador ejecuta y el señalizado continúa más tarde (se introduce en la cola de entrada del monitor). Es el estándar y el que usamos por defecto.
- **SW** (signal and wait): el señalizador espera y el señalizado ejecuta
- **SC** no es *preemptive* (no es interrumpido): el proceso que ejecuta signal retiene el control exclusivo del monitor y el despertado ejecutará más tarde
- **SW** es *preemptive*: el proceso despertado interrumple al señalizador (le quita el procesador)

Operaciones adicionales sobre variables condicionales:

- **Priority wait**: **wait(cv,rank)** donde cv es una variable condicional y rank es una expresión evaluable a un entero que permite que los procesos se retrasen en orden ascendente de rank (en caso de empate se despierta el que ha esperado más)

  mirank(cv): devuelve rank del proceso en la cabeza de la cola
- **Broadcast signal**: cuando más de un proceso atrasado puede proceder o cuando el señalizador no sabe cual debe preceder

  signal_all(cv): while (!empty(cv)) signal(cv);

  SC: bien definido

  SW: no está bien definido si no hay procesos delayed, si hay varios delayed?

Desarrollamos soluciones basadas en monitores para 5 problemas básicos:

- Bounded buffers
- Readers and writers
- Distribución de recursos y planificación
- Temporizador de intervalos
- Sleeping barber