Clase 5

Monitores

Son estructuras de datos destinadas a ser usadas sin peligro por más de un proceso en ejecución. La característica principal es que sus métodos son ejecutados con exclusión mutua. Lo que significa que en cada momento en el tiempo, un profeso como máximo puede estar ejecutando cualquiera de sus procesos.

A diferencia de los semáforos, proporcionarán un mecanismo de abstracción de datos que encapsula datos y operaciones que se pueden hacer sobre estos datos.

Un monitor define un conjunto de operaciones que son los únicos medios para manipular los recursos compartidos. Dentro de un monitor, se pueden definir variables de estado que almacenan información sobre el recurso compartido y el estado actual de los procesos que lo utilizan. Los procedimientos implementan las operaciones que se pueden realizar en el recurso, y estos procedimientos pueden acceder y modificar las variables de estado de manera segura.

Los monitores también admiten mecanismos de comunicación y sincronización, como señales y condiciones de espera, que permiten a los procesos coordinarse y comunicarse de manera efectiva dentro del monitor.

- Exclusión mutua: implícita asegurando que los procesos en el mismo monitor no se ejecuten concurrentemente.
- Sincronización por condición: explícita con variables condición.

La ventaja de los monitores es que un proceso que invoca un procedure puede ignorar cómo está implementado.

Notación

Un monitor tendrá

- Interfaz: específica operaciones que brinda el recurso.
- Cuerpo: tiene variables que representan el estado del recurso y procedures que implementan las operaciones de la interfaz.

Solo los nombres de los procedures son visibles desde afuera. Sintácticamente, los llamados al monitor tienen la forma `NombreMonitor.op_i(argumentos)`.

Los procedures pueden acceder solo a variables permanentes, sus variables locales y parámetros que le sean pasados en la invocación.

```
monitor NombreMonitor
{
  declaraciones de variables permanentes;
  código de inicialización
procedure op1 (par. formales1)
{
  cuerpo de op1
}
.......
procedure op_n (param. formales)
{
  cuerpo de op
}
}
```

La sincronización por condición se implementa mediante uso de variables condición. Son usadas junto con monitores y permiten a los procesos esperar hasta que se cumpla cierta condición antes de continuar su ejecución.

- Variable condición `cv`: se usa para indicar una condición específica que debe cumplirse para que un proceso pueda continuar su ejecución. El valor asociado a `cv` no es visible directamente al programador y suele ser una cola de procesos que están esperando que se cumpla una condición.
- Operaciones en variables condición
  - `wait(cv)` cuando un proceso llama esta variable, se coloca al final de la cola asociada a la variable condición `cv`. Luego, el proceso libera el acceso exclusivo al monitor que estaba utilizando. El proceso se bloquea y espera a que se le despierte cuando la condición se cumpla.
  - `signal(cv)` se utiliza para despertar al proceso que está en la parte frontal de la cola asociada a la variable condición `cv`.
  - `signal_all(cv)`: La operación `signal_all(cv)` despierta a todos los procesos que están esperando en la cola asociada a la variable condición `cv`. Esto vacía

completamente la cola y permite que todos los procesos intenten adquirir el acceso exclusivo al monitor.

- Disciplinas de señalización
  - Signal and continue: disciplina de señalización utilizada en la materia. Cuando se llama a `signal(cv)`, el proceso despertado no retoma automáticamente la ejecución del proceso que hizo la señalización, sino que compite por el acceso al monitor junto con otros procesos.
  - Signal and Wait: cuando se llama a `signal(cv)`, el proceso despertado retoma automáticamente la ejecución del proceso que hizo la señalización. Esto puede llevar a un comportamiento más predecible y es útil en ciertos casos.

Diferencias en disciplinas de señalización

- Signal and Continue: cuando un proceso llama a `signal(cv)` para despertar a otro proceso que está esperando en una variable condición (`cv`), el proceso que hizo la señalización continúa usando el monitor o recurso compartido sin esperar al proceso despertado.

  El proceso despertado se coloca en una cola de espera para competir por acceder nuevamente al monitor. El proceso despertado debe intentar adquirir el acceso al monitor, pero debe competir con otros procesos que también desean acceder.

- Signal and Wait: cuando un proceso llama a `signal(cv)` para despertar a otro proceso que está esperando en una variable condición (`cv`), el proceso que hizo la señalización también compite por acceder nuevamente al monitor. Sin embargo, a diferencia de "Signal and Continue", el proceso despertado retoma automáticamente la ejecución dentro del monitor desde la instrucción que lógicamente sigue a la operación `wait(cv)`. Esto significa que el proceso despertado no necesita competir nuevamente por el acceso al monitor, sino que continúa ejecutándose dentro del monitor desde donde se detuvo.